Экзаменационная работа состоит из двух задач. Не ожидается, что решения задач будут объёмными (мои реализации занимают меньше 20 строк каждая), но нужно проявить некоторую сообразительность, чтобы готовый код отрабатывал за секунды, а не за часы или сутки.

Правила оценивания:

- Одна решённая задача оценивается из максимума 8 баллов.
  Если студент хорошо справился с одной задачей и неплохо выполнил лабораторные работы, я оставляю за собой право поднять оценку до 9 баллов.
- Две решённые задачи оцениваются из максимума 10 баллов.
  Если студент хорошо справился с двумя задачами и неплохо выполнил лабораторные работы, я оставляю за собой право поднять оценку до 11 баллов.

Экзамен открыт до 23:59 воскресенья, 9 июля. Работы, присланные в последний час, т. е. после одиннадцати вечера 9 июля, я вряд ли успею проверить, поэтому желательно не дотягивать до последнего момента. В течение двух недель, отведённых на экзамен, я постараюсь максимально оперативно комментировать присылаемые работы и параллельно понемногу проверять лабораторные.

Задача 1

Рассмотрим такое преобразование f натурального числа n:

$$f(n) = \begin{cases} n/2, & n \text{ чётное;} \\ 3n+1, & \text{иначе.} \end{cases}$$

Гипотеза Коллатца утверждает, что любое натуральное число превратится в единицу, если к нему достаточно много раз последовательно применить преобразование f. Вот пример – сколько раз нужно применить f, чтобы превратить в единицу первые несколько натуральных чисел:

1: 0 раз (это и так единица);
2: 1 раз ($2 \to 1$);
3: 7 раз ($3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$);
4: 2 раза ($4 \to 2 \to 1$);
5: 5 раз ($5 \to 16 \to 8 \to 4 \to 2 \to 1$).

Наименьшее количество применений f такое, что в результате число n превращается в единицу, будем называть длиной траектории числа n по действию f. В нашем примере число 3 имеет самую длинную траекторию, её длина 7.

Напишите функцию `collatz` с одним аргументом n. Функция должна возвращать 2-элементный кортеж:

– первый элемент – число k ($1 \le k < n$) такое, что у него самая длинная траектория среди всех чисел в диапазоне $[1, n)$;
– второй элемент – длина его траектории.

Может быть так, что подходящих чисел несколько: например, в диапазоне $[1, 25)$ самые длинные траектории у чисел $k = 18$ и $k = 19$. Тогда можно выбрать для возвращения любое из них.

На проверку присылайте файл исходного кода `task1.py` с вашей реализацией функции `collatz`. Можете воспользоваться заготовкой `task1_template.py`; в этом случае сначала

добейтесь, чтобы проходили доктесты. Самый тяжёлый тест, $n = 10^6$, должен отрабатывать не дольше, чем за несколько секунд.

Советы по реализации:

- Если вычислять длину траектории каждого числа по отдельности и не запоминать промежуточные результаты, то функция будет работать слишком медленно.
- Может быть полезно написать одну или несколько вспомогательных функций.
- Метод `items` у словаря позволяет итерировать по парам «ключ–значение».

Задача 2

Два слова называются анаграммами, если их буквенный состав одинаков, но буквы идут в различном порядке. Например, `"ватерполистка"` и `"австралопитек"` – анаграммы.

Набор из двух или более слов назовём семейством анаграмм, если любые два слова из него являются анаграммами. Например, `"товар"`, `"отвар"`, `"тавро"` ‘клеймо’ и `"втора"` ‘вторая скрипка в квартете’ – семейство анаграмм.

Файл `zaliznyak.txt` содержит начальные формы всех русских слов, зафиксированных в грамматическом словаре А. А. Зализняка. Это чуть меньше 100 тысяч слов. В файле на каждой строке дано одно слово, записанное кириллическими буквами в нижнем регистре. Кодировка файла – Windows-1251.

Напишите функцию `anagrams` с двумя параметрами: имя входного файла `filein` и имя выходного файла `fileout`. Функция должна считывать слова из входного файла (можно предполагать, что его формат всегда будет таким же, как у `zaliznyak.txt`) и находить семейства анаграмм, максимальные по включению. Когда говорят «максимальные по включению», имеют в виду, что каждое найденное семейство должно быть настолько большим, насколько это возможно. Например, семейство из четырёх слов `"товар"`, `"отвар"`, `"тавро"`, `"втора"` не является максимальным по включению, потому что в словаре Зализняка есть ещё и другие слова, которые можно было бы в него добавить.

Функция должна выводить найденные семейства анаграмм в выходной файл `fileout`. Формат вывода: кодировка Windows-1251, одно семейство на каждой строке, слова в семействе разделены пробелом. Порядок перечисления семейств и слов внутри каждого семейства может быть любым, но для упрощения проверки желательно соблюдать следующие правила:

– слова внутри семейства сортируются лексикографически по возрастанию;
– семейства сортируются в первую очередь по убыванию количества слов, во вторую очередь – лексикографически по начальному слову по возрастанию.

На проверку присылайте файл исходного кода `task2.py` с вашей реализацией функции `anagrams`. Можете воспользоваться заготовкой `task2_template.py`; в этом случае сначала добейтесь, чтобы прописанный в ней вызов `anagrams("zaliznyak.txt", "anagrams.txt")` отрабатывал не дольше, чем за несколько секунд. Доктестов нет, но для справки сообщу, что мой вариант скрипта нашёл 2706 семейств анаграмм в словаре Зализняка.

Советы по реализации:

- Если сравнивать каждое слово с каждым, функция будет работать слишком медленно.
- Подумайте, как использовать встроенную функцию `sorted`.