A. Поиск слова. 2021

Ограничение времени	4.5 секунд
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

По кругу записано несколько букв (возможно, повторяющихся). Студент 317 группы хочет узнать, сможет ли он прочитать некоторое слово, если будет двигаться по кругу (в любом направлении), не пропуская буквы. Студент сам выбирает место, с которого он начинает читать, и направление. Необходимо написать функцию `find_word_in_circle(circle, word)`, которая должна возвращать позицию, с которой нужно начинать чтение (индекс в строке) и направление чтения (1 если слева направо или -1 если справа налево), если студент может найти строку `word` в круговой строке `circle`, и -1 иначе.
Строка `word` содержит как минимум один символ.

Формат ввода

```
find_word_in_circle('napo', 'ap')
```

Формат вывода

1 1

Примечания

На проверку необходимо отправить файл `find_word_in_circle.py` с функцией `find_word_in_circle(circle, word)`

B. Максимальная подстрока. 2021

Ограничение времени	4.5 секунд
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Написать функцию `find_max_substring_occurrence(input_string)`, принимающую на вход непустую строку `input_string`. Функция должна возвращать наибольшее число `k`, такое что `input_string` совпадает с некоторой своей подстрокой `t`, выписанной `k` раз подряд.

Формат ввода

find_max_substring_occurrence('abab')

Формат вывода

2

Примечания

На проверку необходимо отправить файл `find_max_substring_occurrence.py` с функцией `find_max_substring_occurrence(input_string)`.

C. Построение словаря. 2021

Ограничение времени	4.5 секунд
Ограничение памяти	64Mb
Ввод	input.txt
Вывод	output.txt

В файле `input_dict_name` находится человеко-драконий словарь. В первой строке словаря записано число слов, к которым есть перевод. Затем на каждой строке словаря располагается слов и один или несколько переводов к нему. Слово и его переводы разделены дефисом, переводы одного слова разделены запятой. Дефис отделён от соседних слов пробельными символами. После запятой ставится пробельный символ.
Функция `get_new_dictionary(input_dict_name, output_dict_name)` должна по человеко-драконьему словарю, находящемуся в `input_dict_name`, построить драконе-человечий словарь и сохранить его в файл с именем `output_dict_name` в аналогичном исходному словарю формате. Словарь должен быть полным, т.е. учитывать всю информацию, которая находилась в исходном словаре. Строки выходного словаря должны быть отсортированы в лексикографическом порядке.

Формат ввода

Файл `'input.txt'`:
5
cat - kosha
dog - soba
good - horo, normo
bad - ploh, uzha
ugly - uzha

Формат вывода

Файл `'output.txt'`:
6
horo - good
kosha - cat
normo - good
ploh - bad
soba - dog
uzha - bad, ugly

Примечания

На проверку необходимо отправить файл `get_new_dictionary.py` с функцией `get_new_dictionary(input_dict_name, output_dict_name)`.

D. Проверка предложений. 2021

Ограничение времени	4.5 секунд
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Напишите функцию check_first_sentence_is_second, принимающую на вход две строки. Каждая строка задаёт предложение. Необходимо проверить, можно ли получить второе предложения из первого с помощью перестановки и удаления слов. Каждая из строк содержит только буквы и пробелы, любые буквенные последовательности разделённые пробелом считаются разными словами. Если можно, функция должна вернуть True, иначе False.

Формат ввода

check_first_sentence_is_second("люк я твой отец", "я отец твой")

Формат вывода

True

Примечания

На проверку необходимо отправить файл check_first_sentence_is_second с функцией check_first_sentence_is_second.