С вопросами по всем задчам седьмого ДЗ можно писать Дианкину Игорю.

Задание 1 (1 балл)

Мнемоника: naive
Поиск подстроки в строке алгоритмом наивного поиска.
Инпут читается из файла **in.txt**, записывается в файл **out.txt**

В файле одна строка, в ней строка и паттерн разделенные пробелом.
В случае отсутствия находок писать пустую строку.

Input	ATGATGATG ATG
Output	0 3 6

Задание 2 (2 балла)
Мнемоника: rabin
Поиск подстроки в строке алгоритмом Рабина-Карпа.
Инпут читается из файла **in.txt**, записывается в файл **out.txt**

В файле одна строка, в ней строка и паттерн разделенные пробелом.
В случае отсутствия находок писать пустую строку.
Алфавит состоит из букв A T G C.

Input	ATGATGATG ATG
Output	0 3 6

Задание 3 (1 балл)
Мнемоника: prefsuf
Префикс функция
Инпут читается из файла **in.txt**, записывается в файл **out.txt**

В файле одна строка, в ней слово, в аутпутный файл необходимо записать все значения суффикс-префикс функции через пробел.

Input	abcabcabc
Output	0 0 0 1 2 3 4 5 6

Задание 4 (2 балла)
Мнемоника: kmp
Поиск подстроки в строке алгоритмом Кнута-Морриса-Пратта.
Инпут читается из файла **in.txt**, записывается в файл **out.txt**

В файле одна строка, в ней строка и паттерн разделенные пробелом.
В случае отсутствия находок писать пустую строку.

Input	abcabcabc abc
Output	0 3 6

Задание 5 (1 балл)
Мнемоника: roll_hash
Инпут читается из файла **in.txt**, записывается в файл **out.txt**
Написать скользящую хэш-функцию (Rolling Hash).
https://ru.wikipedia.org/wiki/Алгоритм_Рабина_—_Карпа

$$hash(p[1..m]) = \left(\sum_{i=1}^{m} p[i] x^{m-i} \right) \bmod q,$$

Хэш функция должна быть такой. Здесь p[i] - это ASKII номер символа, в python его можно получить функцией ord(). В задании требуется продемонстрировать свойство скольжения хэш функции по данному тексту.
Входные денные: Первая строка входных данных это параметры m, x и q. m - ширина окна, как в рабине-карпе, x и q как в формуле выше. Вторая строка, текст, по подстрокам длины m, которого нужно скользить вашей хэш-функцией.
Выходные данные: Значения хэш функции для каждого окна.
Примечание: На прямую брать хэш от подстроки можно только на первом шаге, все последующие ***должны*** быть рассчитаны через сдвиг.

Input	2 2 32 ABCD
Output	4 7 10

Input	2 2 2 ABCD
Output	0 1 0

Input	3 10 1000 ABCD
Output	227 338

Задание 6 (2 балла)
Мнемоника: bit_search
Инпут читается из файла **in.txt**, записывается в файл **out.txt**

В данном задании при каждом упоминании хэш-функции будет подразумеваться хэш функция написанная вами для предыдущего задания, с параметрами x = 10 и q = 64.

Вам нужно создать 64-битный массив из нулей, как в Фильтре Блума (битовый массив - массив из нулей и единиц).
Вам подается и несколько строк. Нужно проверить какие из строк есть в тексте, взять значения хэш-функции от этих строк, и по значению хэш-функции заменить 0 на 1 в массиве.
Затем полученный массив представить в виде двоичного числа, перевести в десятичную систему счисления и записать в аутпут.
Пример теоретический: У вас есть некий текст, и строка из одного символа “@”. Ваш алгоритм находит @ в тексте, применяет хэш-функцию и получает 63 (Описанная хэш-функция от @ должна давать ноль), отправляется по индексу в последнюю ячейку вашего массива и меняет его на единицу. Массив 0000 … 001 представляя как двоичное сило вы получаете 1, переводите в десятичное.

Input	abbans@hhh @
Output	1

В самом задании встречаются только строки из символов “A” и “B”.

Input	ABBAB ABB BBBB
Output	32

Input	ABABA BAB BBBB
Output	8796093022208

Доп. Задание 7 (2.5 балла)
Мнемоника: palindrome
Реализовать поиск палиндромов с использованием хэширования. Вам подается строка и требуется найти количество всех подстрок-палиндромов в ней. Подстрока может быть равна самой строке.
Решение должно работать за O(nlogn)

Input	AB

Output	0

Input	ABABA
Output	4

Доп. Задание 8 (3 балла)
Мнемоника: kmp_binary

Реализовать Рабина-Карпа для нуклеотидного алфавита с использованием бинарного представления нуклеотидов и бинарных операций (задача, которую давал на подумать Миронов на лекции) для паттерна до длины 16, с возможным несовпадением паттерна и текста на указанное число символов.
Алгоритм должен работать за O(N + M).

Бинарные операции в Python - https://data-flair.training/blogs/python-bitwise-operators/

Указания (смотря указания вы лишаете себя части удовольствия в решении задачи):

1. Для перевода кодирования паттерна полезно использовать операцию &
2. Для смещения позиции понадобятся операции побитового сдвига (>>, <<)
3. Для сравнения паттерна и текущего куска текста понадобится операция xor (^)
4. Чтобы подсчитать число несовпавших нуклеотидов нельзя просто проверять в цикле операцией & количество несовпавших битов - это приведет к ассимптотике O(MN). Необходимо использовать битовые трюки, например, этот. (https://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetTable). Можете посмотреть вот здесь - https://en.wikipedia.org/wiki/Hamming_weight
5. Так как каждый нуклеотид кодируется последовательностью из двух битов, то строго говоря надо считать число не битов, а пар 01, 10 и 11 в числе. Я специально не пишу какие пары и как брать - предыдущих четырех подсказок более чем достаточно.

Input
На первой строке - паттерн длины до 16.
Во второй строке - строка, в которой надо осуществлять поиск. Ее длина не ограничена.
В третьей строке - сколько несовпадений в паттерне и тексте разрешается.
Output
Начала вхождения всех подстрок, отличающихся от паттерна не более чем на разрешенное число совпадений. Если вхождений нет, то пустая строка.

Input	atgc atgcaatgaatggatgcaattt 0
Output	0 13

Input	atgc atgcaatgaatggatgcaattt 0
Output	0 13

Input	atgc atgcaatgaatggatgcaattt 1
Output	0 5 9 13

Input	atgc atgcaatgaatggatgcaattt 2

Output	0 5 9 13 18