#92: ТАБЛИЦА УМНОЖЕНИЯ

Сложность: 1 из 10

Написать программу **mult.py**, которая получает из первого аргумента командной строки число, а после печатает таблицу умножения для этого числа.

Пример использования

```
$ python mult.py 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
```

#40: ПРЯМОЙ И ОБРАТНЫЙ ПОРЯДОК

Сложность: 1 из 10

Написать программу **reverse.py**, которая получает из первого аргумента командной строки текст и меняет в нем порядок букв в словах на обратный. Порядок слов должен быть сохранен.

Пример использования

```
$ python reverse.py "яблоки вкусные"
иколбя еынсукв
```

#28: СУММА ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

Сложность: 1 из 10

В файле **numbers.txt** в одну строку через пробелы записаны целые числа. Нужно написать программу **minus.py**, которая считает и печатает сумму всех отрицательных чисел из файла.

Пример файла numbers.txt

```
1 2 -3 4 -5 6 7
```

Пример использования

```
$ python minus.py
-8
```

#25: ПАЛИНДРОМ

Сложность: 1 из 10

Написать программу **palindrome.py**, которая получает из первого аргумента командной строки число, а затем выводит:

1 — если число является палиндромом,

0 — если не является.

Палиндром — одинаково читающееся в обоих направлениях слово, число или фраза. Например, 8228, ABBA.Подробнее о палиндроме в Википедии.

Пример использования

```
$ python palindrome.py 1881
$ 1
$ python palindrome.py 174
$ 0
```

#10: ТОЛЬКО ГЛАСНЫЕ

Сложность: 2 из 10

Написать программу **vowel.py**, которая получает из первого аргумента командной строки слово на английском языке, удаляет из него все согласные буквы и выводит оставшиеся гласные.

Пример использования

```
$ python vowel.py programming
$ oai
```

#58: СЛЕД МАТРИЦЫ

Сложность: 2 из 10

В файле **matrix.txt** содержится квадратная матрица N x N. Напишите программу **sled.py**, которая вычисляет след матрицы и выводит его на экран.

Пример файла matrix.txt:

```
1 2

3 4
```

Пример использования:

```
$ python sled.py
5
```

#46: ПРОВЕРКА СКОБОК

Сложность: 2 из 10

Написать программу **skobki.py**, которая получает из первого аргумента командной строки строку, содержащую скобки, и проверяет правильно ли они расставлены.

Рассматривать только круглые скобки.

Программа должна выводить:

- **1** – если скобки расставлены правильно;
- **0** – если порядок скобок нарушен.

Пример использования

```
$ python skobki.py "(2+3)()"
1
```

#71: НЕДЕЛЯ ГОДА

Сложность: 3 из 10

Написать программу **week_of_year.py**, которая вычисляет номер недели в году по номеру дня.

Первая неделя года определяется как неделя, содержащая первый понедельник года. Так в 2016 году, первый понедельник пришелся на 4 число, поэтому 1, 2 и 3 числа относятся к неделе предыдущего года, а первая неделя считается с 4 по 10 число включительно.

Программа **week_of_year.py** принимает два параметра: число, с которого началась первая неделя и номер дня в году, а после возвращает номер недели для переданного дня. В случае если неделя относится к предыдущему году, то выводить -1.

Пример использования

```
$ python week_of_year.py 4 4
1
$ python week_of_year.py 4 3
-1
$ python week_of_year.py 4 11
2
```

#80: СВЕТОФОР

Сложность: 3 из 10

Работа светофора для пешеходов запрограммирована следующим образом, с начала каждого часа, в течение N минут горит зеленый сигнал, затем в течение M минут горит красный сигнал. Потом снова N минут горит зеленый и тд.

Написать программу **traffic_light.py**, которая получает три аргумента из командной строки: N, M и T, где T время в минутах, прошедшее с начала очередного часа. После программа должна вывести **green** или **red** в зависимости от того, сигнал какого цвета горит для пешехода в этот момент.

Пример использования

```
$ python traffic_light.py 4 2 3
green
$ python traffic_light.py 4 2 4
red
```

#44: СЛОЖНЫЕ ПРЕДЛОЖЕНИЯ

Сложность: 3 из 10

Дан файл **text.txt**, в котором записан текст. Нужно найти в этом тексте предложения, содержащие больше двух запятых или длина которых превышает 120 символов. Предложения заканчиваются знаком препинания **(.?!)** и пробелом или переводом строки. Найденные предложения нужно вывести в файл **complex.txt** — каждое предложение с новой строки.

Имя программы — **complex.py**

#31: ВОЗРАСТАЮЩАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ

Сложность: 3 из 10

Написать программу **sq.py**, которая получает из первого аргумента командной строки имя файла, а затем выводит на экран:

- **1** – если в файле записана возрастающая последовательность целых чисел;
- **0** – если последовательность не возрастающая.

Каждый элемент последовательности в файле записывается с новой строки.

Если внутри возрастающей последовательности встречаются одинаковые элементы идущие друг за другом, то последовательность в целом считается возрастающей.

Пример файла sq1.txt с возрастающей последовательностью

```
1

2

3

3

```

```
4
```

Пример запуска программы

```
$ python sq.py sq1.txt

1
```

#88: СВЕТОФОР - 2

Сложность: 4 из 10

Работа светофора для автомобилей запрограммирована следующим образом, с начала каждого часа, в течение N минут горит зеленый сигнал, затем в течение M минут горит желтый сигнал, а потом в течение L минут — красный. После снова загорается зеленый и тд.

Написать программу **traffic_light.py**, которая получает четыре аргумента из командной строки: N, M, L и T, где T время в минутах, прошедшее с начала очередного часа. После программа должна вывести **green**, **yellow** или**red** в зависимости от того, сигнал какого цвета горит для автомобиля в этот момент.

Пример использования

```
$ python traffic_light.py 3 1 2 2
green
```

#50: ПЛОТНОСТЬ СЛОВ

Сложность: 4 из 10

Написать программу **stats.py**, которая считает сколько раз в тексте встречаются слова. Текст находится в файле **text.txt**.

Также рядом с программой лежит файл **zn.txt**, который содержит знаки препинания, встречающиеся в тексте. Каждый знак записан с новой строки.

После запуска программа должна создавать файл **stats.txt** с перечислением всех слов текста в алфавитном порядке. После слова идет двоеточие, пробел и число — сколько раз это слово встречается в тексте.

Пример файла text.txt

```
Данное чтиво вряд-ли предназначено для широкого круга
специалистов.

Слишком узок предмет. Слишком большое пересечение технологий и
бизнеса.
```

```
Слишком сложен продукт.
```

Пример файла zn.txt

```
.

?

,

!
```

Пример файла stats.txt

```
Данное: 1

Слишком: 3

бизнеса: 1

большое: 1

вряд-ли: 1

для: 1

и: 1

круга: 1

пересечение: 1

предмет: 1

предназначено: 1

продукт: 1

сложен: 1

специалистов: 1

технологий: 1
```

```
узок: 1

чтиво: 1

широкого: 1
```

Пример использования

```
$ stats.py
```

#33: САМЫЙ ПРОДОЛЖИТЕЛЬНЫЙ ТРЕНД

Сложность: 5 из 10

Написать программу **long_trend.py**, которая получает в виде строки набор целых чисел разделенных пробелами, вычисляет самый продолжительных тренд из этих чисел и выводит его на экран.

Одно и то же число может попадать сразу в два тренда. Например, из последовательности 1 2 3 4 3 2 можно выделить два тренда 1 2 3 4 и 4 3 2 — 4 попадает сразу в оба тренда.

Пример использования

```
$ python long_trend.py "1 2 3 4 3 2"
1 2 3 4
```

#56: ТРАНСПОНИРУЕМ МАТРИЦУ

Сложность: 7 из 10

В файле **matrix.txt** содержится матрица M x N. Напишите программу **transpose.py**, которая транспонирует матрицу и сохраняет результат в файл **transpose_matrix.txt**.

Пример файла matrix.txt:

```
1 2

3 4

5 6
```

Пример файла transpose_matrix.txt (после транспонирования матрицы из matrix.txt):

```
1 3 5

2 4 6
```

Пример использования:

```
$ python transpose.py
```