# Рекурсия

# Длина списка

Kлассная работа

макс. 1 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Рекурсивно вычислите длину списка. Для этого напишите функцию **recursive\_len(some\_list)**, принимающую в себя список и возвращающую его длину.

## Пример

| Ввод | Вывод |
| --- | --- |
| print(recursive\_len([1, 2, 3])) | 3 |

# косипс ьтунревереП

Kлассная работа

макс. 1 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

А теперь попробуйте рекурсивно перевернуть список. Для этого напишите функцию **recursive\_reverse(some\_list)**.

# Трибоначчи

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Оказывается, что в дополнение к ряду Фибоначчи существует ещё и ряд Трибоначчи, который начианется с тройки **0, 0, 1**, а каждое следующее число равно сумме **трёх предыдущих**. Числа нумеруются с 0.

Напишите функцию **tribonacci(n)**, которая принимает в себя номер числа и возвращает n-ое число Трибоначчи.

## Пример

| Ввод | Вывод |
| --- | --- |
| print(tribonacci(0)) | 0 |

## Примечания

Функция должна быть рекурсивной.

# Сумма элементов списка

Kлассная работа

макс. 2 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Мы уже немного умеем писать рекурсивные программы. Давайте тренироваться дальше.  
Напишите рекурсивную функцию **rec\_linear\_sum(some\_list)**, которая вычисляет сумму элементов переданного в неё линейного списка вещественных чисел, то есть такого списка, в котором нет вложений.

## Пример

| Ввод | Вывод |
| --- | --- |
| print(rec\_linear\_sum([])) | 0 |

# Линеаризация списка

Kлассная работа

макс. 3 балл.

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Как вы уже знаете, списки могут иметь разную вложенность, поэтому иногда возникает задача «линеаризовать» список, то есть превратить список с произвольным количеством уровней в линейный.  
  
Как раз это вам и нужно сделать в данной задаче, причём, конечно же, рекурсивно.  
  
Напишите функцию **linear(some\_list)**, которая получает список произвольной вложенности, линеаризирует его и возвращает результат.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| print(\*linear([])) |  |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| print(\*linear([[1, 2], 4, [[2, 4, 8, [-4, "er", [0], [{2: "primer"}], []]]]])) | 1 2 4 2 4 8 -4 er 0 {2: 'primer'} |