Синтаксис Python. Цикл `while`.

Во всех задачах листка нельзя пользоваться операцией возведения в степень (`**`), кроме случаев, когда требуется вычисление квадратного корня.

A. *Факториал*
Дано $n \geqslant 0$, вычислить $n!$

Input	Output
1	1

B. *Сумма цифр числа*
Найти сумму цифр натурального числа.

Input	Output
12345	15

C. *Степень числа*
Дано целое число a и целое неотрицательное число n. Вычислить a^n.

Input	Output
2 3	8

D. *Число Фибоначчи*
Найти n-е число Фибоначчи ($\phi_0 = 0, \phi_1 = 1, \phi_n = \phi_{n-1} + \phi_{n-2}$).
На вход программе даётся одно целое неотрицательное число n.
Программа должна вывести ϕ_n.

Подсказка: на Python "перебор" чисел Фибоначчи можно организовать при помощи т.н. множественного оператора присваивания (обновляем значения нескольких переменных), например: `a, b = b, a`
В левой части оператора присваивания несколько переменных через запятую, а в правой столько же выражений. Эти выражения вычисляются (со "старыми" значениями переменных), а потом полученные значения синхронно присваиваются переменным.

Придумайте, что должно быть написано справа в таком присваивании, чтобы его можно было использовать для вычисления чисел Фибоначчи.

Input	Output
6	8

E. *Наименьший натуральный делитель*
Дано целое число N, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1. Время работы программы должно быть пропорционально $\sqrt{N}$.

Input	Output
15	3

F. *Точная степень двойки*
Дано натуральное число N. Выведите слово `YES`, если число N является точной степенью двойки, или слово `NO` в противном случае.
Операцией возведения в степень пользоваться нельзя. В этой задаче можно обойтись одним циклом и одним условным оператором. Условный оператор разрешается использовать только вне цикла.

Input	Output
1	YES

G. *Двоичный логарифм*
По данному натуральному числу N выведите такое наименьшее целое неотрицательное число k, что $2^k \geqslant N$.
Операцией возведения в степень пользоваться нельзя.

Input	Output
1	0

H. *Номер числа Фибоначчи*

Последовательность Фибоначчи определяется так:

$$\phi_0 = 0, \phi_1 = 1, \phi_2 = 1, \ldots, \phi_N = \phi_{N-1} + \phi_{N-2}$$

Дано натуральное число A. Определите, каким по счету числом Фибоначчи оно является, то есть выведите такое число n, что $\phi_n = A$.
Если A не является числом Фибоначчи, выведите число -1.
Можно расчитывать, что число 1 на вход подаваться не будет.

Напишите цикл, используя такие переменные и инвариант:

- переменные: `k` — номер текущего числа Фибоначчи, `f1`, `f2` — пара соседних чисел Фибоначчи;
- начальные значения переменных:

  ```
  k = 0
  f1 = 0
  f2 = 1
  ```

- инвариант: число Фибоначчи с номером `k` равно `f1`

Input	Output
8	6

I. *Алгоритм Евклида*

Алгоритм Евклида для вычисления наибольшего общего делителя (gcd — greatest common divisor) двух натуральных чисел a и b ($a > b$) основан на следующих трёх фактах (для любых натуральных a и b):

$$gcd(a, b) = gcd(b, a)$$

$$gcd(a, 0) = a$$

$$gcd(a, b) = gcd(a - b, b)$$

Можно сообразить, что многократно выполняемую операцию вычитания можно заменить операцией взятия остатка от деления:

$$gcd(a, b) = gcd(b, a\%b)$$

Рекомендуется выполнить соответствующие вычисления для таких, например, пар чисел: $a = 39, b = 11$ **и** $a = 14, b = 52$**.**
Обратите внимание на то, как изменяются числа в указанном выше соотношении, если первое число меньше второго. Например, чему равен остаток $14\%52$**?**
Если вы уловили идею, то условный оператор `if`, а также функции `max` и `min` вам не понадобятся.

Напишите программу, которая вычисляет НОД двух данных натуральных чисел, не превосходящих 10^{100}.
На вход программе подаются два натуральных числа: a и b.
Программа должна вывести одно число — наибольший общий делитель двух введённых чисел.

Input	Output
21 14	7

J. *Исполнитель "Раздвоитель"*

Исполнитель «Раздвоитель» преобразует натуральные числа. У него есть две команды: “Вычесть 1” и “Разделить на 2”, первая команда уменьшает число на 1, вторая команда уменьшает число в два раза, если оно чётное, иначе происходит ошибка.

Дано два натуральных числа A и B ($A > B$). Напишите алгоритм для Раздвоителя, который преобразует число A в число B и при этом содержит минимальное число команд. Команды алгоритма нужно выводить по одной в строке, первая команда обозначается, как `-1`, вторая команда как `:2`.

Input	Output
179 20	-1 :2 -1 :2 :2 -1 -1

K. *Отрезок на клетчатой бумаге*

Петя нарисовал на клетчатой бумаге отрезок из точки с координатами (a, b) в точку с координатами (c, d). Через сколько клеток проходит этот отрезок?

Считается, что отрезок проходит через клетку, если он проходит через её внутренность. Если же он проходит только через вершину или по границе клетки, считается, что он не проходит через клетку.

Вводятся целые числа a, b, c, d. Числа по модулю не превышают 10^9.

Выведите одно число — количество клеток, через которые проходит отрезок.

Input	Output
0 0 6 4	8
3 3 -3 3	0

L. *Количество целочисленных точек в круге*

Дано натуральное число $R \leqslant 10^5$. Определите количество целочисленных точек, находящихся внутри и на границе круга радиуса R с центром в начале координат.

Сложность алгоритма должна быть $O(R)$.

При решении этой задачи нельзя пользоваться функцией вычисления квадратного корня.

Input	Output
2	13

M. *Быстрое возведение в степень*

Решите задачу C для возведения вещественного числа в неотрицательную степень, используя следующие тождества:

$$x^{2k} = (x^2)^k$$
$$x^{2k+1} = x^{2k} \cdot x$$

На вход программе даётся два числа, каждое в отдельной строке. Первое число — вещественное a, второе целое неотрицательное N.

Программа должна вывести a^N.

Сформулируйте цикл, поддерживающий следующий инвариант:

$$x^k \cdot result = a^N$$

Начальные значения введённых переменных следующие: $x = a, k = N, result = 1$.

Если вы всё сделаете правильно, программа сможет возводить в степень очень быстро, например, для возведения в степень $N = 1000000$ потребуется не более 40 умножений.

Вещественные числа считываются при помощи конструкции `x = float(input())`.

Input	Output
2 1	2

N. *Разложение на простые множители*

Выписать разложение числа N ($2 \leqslant N \leqslant 2 \cdot 10^9$) на простые множители.

Сложность алгоритма $O(\sqrt{N})$.

Input	Output
18	2 3 3

O. *Расширенный алгоритм Евклида*

Даны два натуральных числа a и b. Найдите их наибольший общий делитель d и *любые* два целых числа x и y, таких, что $ax + by = d$. Программа должна вывести числа d, x, y.

Указание: рассмотрите систему уравнений

$$\begin{cases} a \cdot 1 + b \cdot 0 = a \\ a \cdot 0 + b \cdot 1 = b \end{cases}$$

Затем для правых частей уравнений вычислять НОД(a, b), используя тождество

$$a\%b = a - b * (a \mathbin{/\!/} b)$$

а для левых частей выполнять соответствующие преобразования над коэффициентами при a и b.

Input	Output
1 1	1 0 1
3 7	1 5 -2
66 48	6 3 -4

P. *Количество чисел с чётной суммой цифр на отрезке*

Подсчитайте количество натуральных чисел на отрезке от A до B, сумма цифр которых чётна.

Вводится два натуральных числа A и $B, 1 \leqslant A \leqslant B \leqslant 10^{18}$.

Выведите одно число — количество натуральных чисел, больших либо равных A и меньших либо равных B, имеющих чётную сумму цифр.

Input	Output
1 5	2

Указание: на *любом* отрезке из 10 чисел, где первое заканчивается нулём, а последнее — девяткой, количество чисел с чётной суммой *всегда* равно 5.

В задачах `Q - Z` последовательность целых чисел вводится с клавиатуры, не может содержать число 0 и заканчивается нулём, который служит сигналом окончания ввода последовательности.
В этих задачах не разрешается запоминать последовательность, если вы вдруг уже знакомы с массивами.
Оформлять решение следует таким образом (с точностью до именования переменных):

```
x = int(input())      # чтение первого числа последовательности
...                   # необходимые предварительные присваивания и т.п.

while x != 0:
    ...               # основной алгоритм, в котором нет ни одного
                      # вызова input()
    x = int(input())  # чтение очередного элемента завершает тело цикла

...                   # печать результата ПОСЛЕ цикла
```

Кроме того, требуется давать используемым переменным осмысленные имена (например, если это длина, текущая длина, максимум/минимум, предыдущий и т.п.)

Q. *Максимум последовательности*
В последовательности целых чисел определить значение наибольшего элемента последовательности.

Input	Output
1 7 9 0	9

R. *Второй максимум*
Последовательность состоит из *различных* целых чисел и завершается числом 0.
Определите значение второго по величине элемента в этой последовательности. Можно считать, что в последовательности не меньше двух элементов.
Второй по величине элемент — максимальный элемент в последовательности, полученной из исходной удалением одного элемента, равного её максимуму.

Input	Output
1 7 9 0	7

S. *Количество элементов, равных максимальному*
Последовательность состоит из целых чисел и завершается числом 0. Определите количество элементов последовательности, равных её максимальному элементу.

Input	Output
1 4 4 3 0	2

T. *Количество перемен знака*
Определить число перемен знака в последовательности целых чисел, заканчивающейся нулём.

Input	Output
-1 -3 -3 3 4 0	1

U. *Сумма чисел между предпоследней и последней двойками*

Последовательность состоит из натуральных чисел и завершается числом 0. Вычислить сумму всех элементов последовательности натуральных чисел между предпоследней и последней двойками (не включая сами двойки). Если двоек нет или она одна, вывести число −1.

Input	Output
3 2 1 2 3 2 1 0	3

V. *Длина наибольшей площадки*

Последовательность состоит из целых чисел и завершается числом 0. Определить длину наибольшей «площадки» в последовательности (т.е. подпоследовательности подряд идущих одинаковых чисел).

Input	Output
1 7 7 9 1 0	2

W. *Длина наибольшего монотонного фрагмента*

Определите наибольшую длину монотонного фрагмента последовательности целых чисел, заканчивающейся нулём (то есть такого фрагмента, где все элементы либо строго больше предыдущего, либо строго меньше).

Input	Output
1 6 7 9 4 1 0	4

X. *Количество локальных максимумов*

Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элемента последовательности. Первый и последний элемент последовательности не являются локальными максимумами.

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите количество строгих локальных максимумов в этой последовательности.

Input	Output
1 2 1 2 1 0	2

Y. *Наименьшее расстояние между локальными максимумами*

Определите наименьшее расстояние между двумя локальными максимумами последовательности. Если в последовательности нет двух локальных максимумов, выведите число 0.

Начальное и конечное значение при этом локальными максимумами не считаются.

Input	Output
1 2 1 1 2 1 2 1 0	2

Z. *Стандартное отклонение*

Дана последовательность натуральных чисел $x_1, x_2, \ldots, x_n$. Стандартным отклонением называется величина:

$$\sqrt{\frac{(x_1 - s)^2 + (x_2 - s)^2 + \cdots + (x_n - s)^2}{n - 1}}$$

где $s = \dfrac{x_1 + x_2 + \cdots + x_n}{n}$ — среднее арифметическое последовательности.

Вычислите стандартное отклонение для данной последовательности натуральных чисел, завершающейся числом 0.

Input	Output
1 7 9 0	4.16333199893