Строки.

Почитать про строки можно здесь: справочник, стр. 15 (глава 3) и стр. 37 (глава 8).
Справочник на английском, описывающий стандартные методы работы со строками: документация по языку Python.
Во всех задачах запрещается использование констант, обозначающих порядковые номера любых символов в таблице `ASCII`.
Для изменения строки лучше пользоваться следующим приёмом. Сперва преобразовать строку в массив отдельных символов: `x = list(s)`, сделать все необходимые изменения (массивы допускают изменения своих элементов в отличие от строк), затем вывести получившуюся строку: `print(''.join(x))`.

A. *Палиндром*
Строка называется *палиндромом*, если она читается слева направо и справо налево одинаково.
Программа должна вывести слово `YES`, если введённое слово — палиндром, и слово `NO`, если оно не является палиндромом.
Решите эту задачу, используя не более $N//2$ операций сравнения символов и не используя сравнений строк и их срезов.

Input	Output
kazak	YES

B. *Количество слов*
Дана строка, содержащая произвольные символы. Определите количество слов в этой строке.
Слово — это несколько подряд идущих букв латинского алфавита (как заглавных, так и строчных), ограниченных слева и справа символами-не-буквами или началом/концом строки.
В примере входных данных пробелы обозначены символом `~`.

Input	Output
Yesterday,~all~my~troubles~seemed~so~far~away	8
hmm,wrong~spaces~~here	4

C. *Самое длинное слово*
Напишите программу, которая выводит самое длинное слово переданной её символьной строки.
Слово — это последовательность символов, отличных от пробела, ограниченная пробелами или концами строки.
Программа должна вывести в первой строке самое длинное слово переданной ей строки, а во второй — длину этого слова. Если слов максимальной длины несколько — вывести первое встретившееся слово максимальной длины.

Input	Output
abra cadabra fibra	cadabra 7

D. *Замена регистра - I*
Дана строка. Напечатать строку, в которой вместо строчных букв исходной строки будут соответствующие прописные и наоборот.

Input	Output
hELLO	Hello

E. *Красные и синие - I*
Дана цепочка, состоящая из синих (`B`) и красных (`R`) точек, всего не более 500000 точек. Требуется выяснить, какое минимальное количество синих точек можно удалить так, чтобы сначала шли только синие, а потом — только красные.
Нужно вывести полученную цепочку, в которой сначала идут только синие точки, а потом — только красные. Во второй строке нужно вывести количество удалённых синих точек.
Здесь полезно не моделировать процесс удаления символов для получения правильного результата, а подсчитать количество букв `R` и `B` в получающейся строке. Потом использовать умножение строки на число и сложение строк.

Input	Output
BBBRBRBRR	BBBRRRR 2

F. *Слова наоборот*

На вход программе подаётся строка, содержащая слова, разделенные пробелами (можно считать, что строка содержит только строчные буквы и пробелы и есть как минимум одно слово).
Программа должна напечатать строку, содержащую те же слова в обратном порядке, которые разделены *одним* пробелом (сами слова не меняются, меняется их порядок).
Напечатанная строка не должна начинаться с пробела или заканчиваться им.
В примере входных данных пробелы обозначены символом `~`.

Input	Output
`~~~~~abcd~~~~~efgh~~~~~~~~prst~~~~`	`prst efgh abcd`

G. *Количество чисел*

Дана строка, содержащая произвольные символы. Посчитать количество *натуральных чисел*, записанных в этой строке.
Натуральное число — последовательность цифр, начинающаяся не с нуля и не являющаяся частью другой последовательности, образующей натуральное число.
Например:
для входной строки `abc123 2023 000340004` программа должна вывести число 3
Комментарий к первому примеру: строки `'0004'`, `'23'`, `'40004'` не являются записью натурального числа в смысле данного выше определения, т.к. содержатся как подстроки в других натуральных числах, соответственно `'340004'`, `'2023'`.

Input	Output
`abc123 2023 000340004`	`3`
`2 0000 0 00`	`1`

H. *Шифр Юлия Цезаря*

Юлий Цезарь использовал следующий способ шифрования текста: каждая буква заменялась на следующую по алфавиту через K позиций по кругу. То есть, например, при $K = 2$ буква `B` заменялась на `D`, буква `X` заменялась на `Z`, в буква `Y` заменялась на `A`.
Вам задана строка длины не превосходящей 10^5, состоящая только из заглавных букв латинского алфавита — результат шифровки, и число K ($0 \leqslant K \leqslant 10^6$), использованное при шифровании.
Необходимо по этим данным определить исходный текст.

Input	Output
`XPSE` `1`	`WORD`

I. *IP-адрес*

IP-адрес это четырёхбайтовый код, который принято записывать в виде четырех десятичных чисел, разделенных точками. Каждое из чисел может принимать значения от 0 до 255. Вот примеры правильных IP-адресов:
`127.0.0.0`
`192.168.0.1`
`255.0.255.255`
Обратите внимание, что числа не могут начинаться с нуля. То есть, такая строка не является IP-адресом:
`127.0.0.01`
Напишите функцию, которая будет возвращать `True`, если переданная строка является правильным IP-адресом, и `False` в противном случае.
На вход программе подаётся произвольная строка. Программа должна вывести строку `YES`, если это правильный IP-адрес и `NO` в противном случае.

Input	Output
`127.0.0.1`	`YES`

J. *Калькулятор*

Напишите программу, которая вычисляет арифметическое выражение, введённое в виде символьной строки. Выражение содержит только целые числа и знаки сложения и вычитания. Гарантируется, что есть как минимум одно число.
Функцию `eval` использовать нельзя.

Input	Output
`1+12-54+68-17`	`10`

K. *Распаковка строки*

Будем рассматривать только строчки, состоящие из заглавных латинских букв. Например, рассмотрим строку `AAAABCCCCCDDDD`. Длина этой строки равна 14. Поскольку строка состоит только из латинских букв, повторяющиеся символы могут быть удалены и заменены числами, определяющими количество повторений. Таким образом, данная строка может быть представлена как `4AB5C4D`. Длина такой строки 7. Описанный метод мы назовём упаковкой строки.

Напишите программу, которая берёт упакованную строчку и восстанавливает по ней исходную строку.

Программа получает на вход одну упакованную строку. В строке могут встречаться только конструкции вида `nA`, где n — количество повторений символа (целое число от 2 до 99), а `A` — заглавная латинская буква, либо конструкции вида `A`, то есть символ без числа, определяющего количество повторений. Максимальная длина строки не превышает 80.

Программа должна вывести восстановленную строку. При этом выведенная строка должна быть разбита на куски длиной ровно по 40 символов (за исключением последней, которая может содержать меньше 40 символов).

Input	Output
AB2C	ABCC
3N34Q12U45A7M24O	NNNQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQUUU UUUUUUUUUAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAMMMMMMMOOOOOOOOOOOOOOOOOOOO OOOOO

L. *Сколько раз входит одна строка в другую*

Даны две строки: `T` и `P`. Найти количество вхождений строки `P` в строку `T`, т.е. количество таких индексов строки `T`, начиная с которых строка `P` входит в строку `T`.

Решите эту задачу без использования дополнительных строк и срезов, а также без сравнения срезов (разрешается сравнивать только отдельные символы).

На вход программе подаётся две строки: в первой строка `T`, во второй строка `P`. Программа должна вывести одно целое неотрицательное число.

Input	Output
AAABAABBBABAABBBABBABA AAB	3
AAAAA AA	4

M. *Форматирование строки*

Дана текстовая строка, содержащая буквы латинского алфавита, пробелы, запятые и точки. Отформатируйте этот текст по следующим правилам:

- В начале и конце строки не должно быть пробелов.
- Все слова разделяются ровно одним пробелом.
- Точки и запятые пишутся слитно с предыдущим словом, после знака препинания ставится пробел (если этот знак препинания не завершает строку).

Выведите полученную строку. После неё обязательно следовать символ перевода строки.

Гарантируется, что никакие два знака препинания не идут подряд.

В примере входных и выходных данных пробелы обозначены символом `~`.

Input	Output
~~Hello~,~world.~~~	Hello,~world.

N. *Из десятичной в римскую*

Напишите программу, которая выводит запись заданного десятичного числа, не превосходящего 3999, в римской системе счисления. Подробнее о правилах перевода здесь (link).

Input	Output
1234	MCCXXXIV

O. *Палиндромы без учёта регистра*

Дано слово, состоящее только из заглавных и строчных латинских букв. Проверьте, является ли это слово палиндромом, если считать заглавные и строчные буквы не различающимися. Выведите слово `YES`, если слово является таким палиндромом и слово `NO`, если не является.

При решении этой задачи нельзя изменять входную строку, пользоваться вспомогательными массивами или строками.

Input	Output
`Radar`	`YES`

P. *Палиндромы без учёта пробелов*

Дана строка, состоящая из строчных латинских букв и пробелов. Проверьте, является ли она палиндромом без учета пробелов (например, `"аргентина манит негра"`).

В этой задаче запрещается изменять входную строку, использовать вспомогательные строки, срезы, а также сравнивать строки, а не отдельные символы.

Input	Output
`ab a`	`YES`

Q. *Палиндромы без учёта гласных букв*

Возьмем произвольное слово и проделаем с ним следующую операцию: поменяем местами его первую согласную букву с последней согласной буквой, вторую согласную букву с предпоследней согласной буквой и т.д. Если после этой операции мы вновь получим исходное слово, то будем называть такое слово негласным палиндромом.

Например, слова `sos`, `rare`, `rotor`, `gong`, `karaoke` являются негласными палиндромами.

Вам требуется написать программу, которая по данному слову определяет, является ли оно негласным палиндромом.

Перечень согласных букв английского алфавита вам предлагается составить самостоятельно.

В этой задаче запрещается изменять входную строку, использовать вспомогательные строки, срезы, а также сравнивать строки, а не отдельные символы.

Input	Output
`tennete`	`YES`

R. *Магическая последовательность*

Даны последовательности: $1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, \ldots$

Выпишите k-ю последовательность.

Можно использовать наивное вычисление этой последовательности.

Начать с первого элемента и $k - 1$ получать следующий.

Input	Output
`4`	`1211`

S. *Majority*

Известно, что в строке один из символов встречается чаще остальных вместе взятых. Вывести этот символ, сделав один проход по строке (считайте, таким образом, что встроенные методы работы со строками тоже под запретом). В качестве дополнительных переменных разрешается использовать только строки длины 1 и целые числа. Исходную строку изменять нельзя.

На вход программе подаётся строка, содержащая не более 10^6 символов.

Напишите доказательство корректности алгоритма.

Input	Output
`abcabcaaaa`	`a`

T. *Замена регистра - II*

Дана строка, содержащая произвольные символы. Измените регистр символов, являющихся буквами латинского алфавита в этой строке так, чтобы первая буква каждого слова была заглавной, а остальные буквы — строчными.

Слово — это последовательность подряд идущих букв, ограниченная слева и справа либо концом исходной строки, либо символами не буквами.

Input	Output
`this is an example`	`This Is An Example`

U. *Красные и синие - II*

Дана цепочка, состоящая из синих (B) и красных (R) точек, состоящая не более, чем из $5 \cdot 10^5$ символов. Нужно удалить наименьшее количество красных точек так, чтобы сначала шли только синие, а потом — только красные.

В этой задаче запрещается использовать вспомогательные строки, срезы, а также сравнивать строки, а не отдельные символы.

Нужно вывести полученную цепочку, в которой сначала идут только синие точки, а потом — только красные. Во второй строке нужно вывести количество удалённых красных точек.

Input	Output
BBBRBRBRRRBBBBRRR	BBBBBBBBBRRR 5

V. *Красные и синие - III*

Дана цепочка, состоящая из синих (B) и красных (R) точек, содержащая не более $5 \cdot 10^5$ символов. Нужно удалить наименьшее одинаковое количество синих и красных точек так, чтобы сначала шли только синие, а потом — только красные.

Выведите полученную цепочку, в которой сначала идут только синие точки, а потом только красные. Во второй строке нужно вывести количество удалённых (синих и красных) точек.

Input	Output
BBBRBBRBRBRRRRRR	BBBBBRRRRRR 4

W. *Идеальные стихи*

Вы когда-нибудь задумывались над тем, как отличить хорошие стихи от посредственных? Редактор литературного журнала занимается этим каждый день, получая тонны корреспонденции от молодых авторов, желающих стать известными поэтами. Благо, в последнее время большая часть стихов присылается по электронной почте, поэтому у редактора возникла мысль автоматизировать процесс. Он твердо уверен, что стихи тем лучше, чем точнее в них рифма. Он считает две строки зарифмованными, если у них совпадает несколько последних букв. И чем больше букв совпадает, тем лучше зарифмованы строки. Например, у строк “палка” и “веревка” совпадают только пары последних букв “ка”, а у строк “олимпиада” и “рая и ада” совпадают четыре буквы (пробелы мы пропускаем). Поэтому вторая рифма лучше.

Редактор считает, что в четверостишии (четыре строки) первая строка должна рифмоваться с третьей, а вторая — с четвертой. Для каждой из этих двух пар строк он считает количество совпадающих последних символов и из этих двух чисел выбирает наибольшее. Полученное число он называет коэффициентом качества стихотворения – чем он выше, тем больше шансов у стихотворения быть опубликованным. Помогите редактору – напишите программу, которая определяет качество стихотворения. И кто знает, может быть, благодаря вашим усилиям, мир познакомится с гениальными стихами (см. первый пример).

Input	Output
yapomnyuchudnoemgnovenje peredomnojyavilasty kakmimoletnoevidenje kakgenijchistoykrasoty	4
eto vovse ne stihi	0
etootlichnyestihi etootlichnyestihi etootlichnyestihi etootlichnyestihi	17

X. *Расшифровка*

Широко известный алгоритм шифрования, т.н. шифр подстановки, заключается в том, что каждый символ исходного сообщения заменяется на другой (одинаковые символы переходят в одинаковые). Такой шифр легко взломать, если частоты букв в исходном сообщении заметно различаются (например, частоты букв `E`, `A`, `O` гораздо больше частот букв `Ж` или `Ю`). Один из способов усложнить задачу взломщику — менять правила замены для каждого символа.

Зашифруем фразу `ANEFFETETOMATOOFMONTANAOFTENATEMONFATTOFFEE`. Для этого выберем *секретное слово*, например, `TRYME`. Припишем его к исходной строке слева, и получившуюся строку обрежем справа так, чтобы её длина стала равна длине исходной строки. Мы получили ключ сообщения. Теперь каждый символ исходной строки циклически сдвинем на величину, соответствующую букве ключа. Буква `A` ключа означает сдвиг на 0 позиций, буква `B` ключа означает сдвиг на 1 позицию и т.д. Ниже приведён пример шифрования, изучите его.

```
ANEFFETETOMATOOFMONTANAOFTENATEMONFATTOFFEE   (исходный текст)
TRYMEANEFFETETOMATOOFMONTANAOFTENATEMONFATT   (ключ)
-------------------------------------------------------------
TECRJEGIYTQTXHCRMHBHFZOBYTRNOYXQBNYEFHBKFXX   (зашифрованное сообщение)
```

Напишите программу, которая по зашифрованному сообщению и секретному слову восстанавливает исходное сообщение.

Input	Output
`TECRJEGIYTQTXHCRMHBHFZOBYTRNOYXQBNYEFHBKFXX` `TRYME`	`ANEFFETETOMATOOFMONTANAOFTENATEMONFATTOFFEE`

Y. *Сделать палиндромом*

Дано слово, состоящее только из строчных латинских букв. Определите, какое наименьшее число букв нужно дописать к этому слову справа так, чтобы оно стало палиндромом.

Input	Output
`abcd`	`3`

Z. *Следующий палиндром*

Рассмотрим все натуральные числа, запись которых в десятичной системе счисления является палиндромом (при этом запись не начинается с нуля). Например, числа 121 и 1331 являются палиндромами, а число 123 — нет. По данному натуральному числу N определите следующее за ним натуральное число (то есть наименьшее число, которое превосходит N), являющееся палиндромом.

Программа получает на вход одно натуральное число N, состоящее не более чем из 200 цифр. Программа должна вывести наименьшее натуральное число, которое больше N и является палиндромом.

Input	Output
`4321`	`4334`