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. *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

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

Input	Output
hELLO	Hello

D. *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

E. *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`
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

F. *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.

Входной файл содержит 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.

В выходной файл выведите восстановленную строку. При этом строка должна быть разбита на строчки длиной ровно по 40 символов (за исключением последней, которая может содержать меньше 40 символов).

Input	Output
`AB2C`	`ABCC`

G. *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`

H. *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`

I. *Магическая последовательность*

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

Выпишите k-ю последовательность.

Input	Output
`4`	`1211`

J. *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.

Input	Output
`abcabcaaaa`	`a`

K. *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`

L. *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