Массивы.

Во всех задачах можно пользоваться любыми встроенными функциями: `max`, `min`, `count` и прочими.
Вводимая строка из слов, разделённых пробелами читается так:

```
x = input().split()
```

Результат — массив строк.

Если нужно только пройтись по этим словам, не создавая переменную для массива слов, можно сразу писать цикл:

```
for word in input().split():
    # do something
```

Строку, в которой записаны целые числа, разделённые пробелом, можно считать так:

```
x = list(map(int, input().split()))
```

Термин ACGT-строка означает строку, состоящую только из символов `A`, `C`, `G`, `T`.

A. *Линейный поиск*

Напишите программу, которая выводит номера элементов массива, равных данному числу.
В первой строке задается одно натуральное число N, не превосходящее 1000 — размер массива.
Во второй строке вводятся N чисел — элементы массива (целые числа, не превосходящие по модулю 1000).
В третьей строке содержится одно целое число x, не превосходящее по модулю 1000.
Вывести номера элементов, равных данному, в порядке возрастания в одну строку через пробел.
Нумерация элементов начинается с единицы. Если таких элементов нет, ничего выводить не нужно.

Input	Output
5 3 -2 5 3 7 3	1 4

B. *Переставить соседние*

Переставьте соседние элементы массива (`A[0]` с `A[1]`, `A[2]` с `A[3]` и т.д.). Если элементов нечётное число, то последний элемент остается на своем месте.
В первой строке вводится натуральное число N, во второй — N целых чисел через пробел.
Программа должна изменить массив в соответствии с условием задачи и вывести его.

Input	Output
5 1 2 3 4 5	2 1 4 3 5

C. *Одна мутация*

В i-м символе ACGT-строки произошла мутация: этот символ надо заменить на другой.
На вход программе даётся три строки: в первой записана ACGT-строка, во второй число — индекс строки k $(0 \leqslant k < len(s))$, где произошла мутация, в третьей строке один символ `A`, `C`, `G` или `T` — на который надо заменить k-й символ данной строки.
Программа должна вывести получившуюся строку.

Input	Output
AACGTACGGACTAGC 4 C	AACGCACGGACTAGC

D. *Наибольшая сумма двух соседних*

Найдите наибольшее значение суммы двух соседних элементов в данном массиве. Можно считать, что в массиве есть не менее двух элементов.
На вход подаётся натуральное число $N(2 \leqslant N \leqslant 10^5)$. Затем на вход подаётся строка, в которой через пробел записаны N целых чисел. Каждое число не превосходит по модулю 10^9.
Программа должна вывести единственное число — ответ на вопрос задачи.

Input	Output
5 1 2 6 -5 4	8

E. *Арифметическая прогрессия*

Дан целочисленный массив. Выяснить, являются элементы данного массива элементами какой-то арифметической прогрессии, идущими подряд? Если да, вывести разность этой арифметической прогрессии, иначе вывести слово `NO`. Можно считать, что массив содержит не менее двух чисел.

Input	Output
5 1 3 5 7 9	2

Input	Output
5 1 2 3 4 6	NO

F. *Ближайшее число*

Напишите программу, которая находит в массиве элемент, самый близкий по величине к данному числу.

Если таких чисел несколько — выведите любое.

На вход подаётся натуральное число $N (2 \leqslant N \leqslant 10^5)$. Затем на вход подаётся строка, в которой через пробел записаны N целых чисел. Каждое число не превосходит по модулю 10^9. В последней строке даётся целое число A.

Программа должна вывести одно из чисел массива — ближайшее к числу A. Если таких несколько — выведите любое.

Input	Output
5 1 2 3 4 5 6	5

G. *Сколько различных элементов в упорядоченном массиве*

Дан массив, упорядоченный по неубыванию элементов в нём. Определите, сколько в нём различных элементов. Размер массива не превышает 10^5.

Input	Output
6 1 2 2 3 3 3	3

H. *Слова наоборот*

Дана строка, состоящая из слов, составленных из латинских букв.

Вывести те же слова, но развёрнутые задом наперёд.

Input	Output
ehal greka cherez reku vidit greka v reke rak	lahe akerg zerehc uker tidiv akerg v eker kar

I. *Извлечь буквы A, C, G, T*

Дана строка, состоящая из слов, разделённых пробелами.

Создайте массив слов, состоящих только из букв `A`, `C`, `G`, `T` имеющихся в словах данной строки.

Если в слове исходной строки нет не одного символа `A`, `C`, `G`, `T`, поместите в массив строку `:|`.

Программа должна вывести массив получившихся строк (без форматирования, см. пример).

Input	Output
DAG CAT BIOLOGY ACGTSTRING TADAM 8E	['AG', 'CAT', 'G', 'ACGTTG', 'TAA', ':|']

J. *Биологические слова*

Дана строка, состоящая из слов, разделённых пробелами.

Выведите только те слова, которые состоят из символов `A`, `C`, `G`, `T`.

Программа должна вывести массив получившихся строк (без форматирования, см. пример).

Input	Output
DAG CAT C BIOLOGY GGGGGGG TADAM 8E	["CAT", "C", "GGGGGGG"]

K. *Биологические слова — много строк*

Решите задачу «*Биологические слова*», если дано несколько строк.

В первой строке на вход программе даётся количество строк N ($1 \leqslant N \leqslant 1000$), затем даётся N строк.

Формат вывода такой же.

Указание: те, кто хочет решить задачу грамотно — реализуйте функцию `collect_words`, принимающую на вход два параметра: строку, содержащую слова и массив с биологическими словами. Функция должна изменять массив.

Input	Output
3 DAG CAT C BIOLOGY GGGGGGG TADAM 8E	["CAT", "C", "GGGGGGG"]

L. *Разбиение на триплеты*

Дана ACGT-строка длины кратной трём. Вывести массив триплетов (без форматирования, см. пример), из которых она составлена.

Указание: если вы вспомните, что такое срез строки, то решение получится и лаконичным и понятным одновременно.

Input	Output
AGCCATGTAGCTAACTCAGGT	["AGC", "CAT", "GTA", "GCT", "AAC", "TCA", "GGT"]

M. *Чётные и нечётные числа*

Реализуйте функцию `split_even_odd`, принимающую на вход один целочисленный массив и возвращающую два массива: первый содержит только чётные числа из исходного, второй — только нечётные числа из исходного массива. Числа в каждом массиве должны следовать в том же порядке, в каком они шли в исходном массиве.

Переданный в функцию массив изменять нельзя.

Функция может возвращать несколько значений, для этого надо в операторе `return` перечислить эти значения через запятую. Например, так: `return even, odd`

Input	Output
even, odd = split_even_odd([11, 202, 37, 40, 57, 63]) print(even) print(odd)	[202, 40] [11, 37, 57, 63]

N. *Выбрать элементы массива*

Напишите функцию `select_elements`, принимающую на вход два массива целых чисел. Функция должна вернуть массив, который содержит элементы первого массива с индексами, перечисленными во втором.

Функция не должна изменять ни один из переданных массивов. Функция должна игнорировать отрицательные индексы, и индексы, превосходящие допустимые для первого переданного массива.

Input	Output
x = [3, -2, 199, 23, -17] print(select_elements(x, [0, 1, 2])) print(select_elements(x, [2, 3, 5])) print(select_elements(x, [-1, 10, 58])) print(select_elements(x, [0, 1, 2, 3, 4, 5]))	[3, -2, 199] [199, 23] [] [3, -2, 199, 23, -17]

O. *Проверка рамки считывания*

Аминокислота кодируется т.н. *триплетом* (или *кодоном*) — ACGT-строкой из трёх символов.
Пептид (последовательность аминокислот) кодируется последовательностью кодонов.
Среди кодонов выделяют *старт-кодон* (`ATG`) и *стоп-кодоны* (`TAA`, `TAG`, `TGA`). Первый указывает на начало рамки считывания пептида, вторые указывают на её конец.
Напишите функцию, которая проверяет есть ли в заданной строке запись пептида. Функция должна называться `is_reading_frame`, принимать на вход одну строку и возвращать булевское значение `True` или `False` в зависимости от ответа.
Будем считать, что рамка считывания в строке есть, если в ней есть старт-кодон и один из стоп-кодонов, причём второй находится правее первого и количество символов между ними отлично от нуля.
Указание 1: учтите, что старт-кодонов и стоп-кодонов в передаваемой строке может быть несколько. Нас интересует *существует ли* хотя бы одна такая пара старт- и стоп-кодона, удовлетворяющая условию задачи?
Указание 2: в методе `s.find()` кроме обязательного аргумента — строки, которую ищем, есть ещё два необязательных параметра, `start` и `end` — они позволяют определить срез строки `s[start:end]`. Если указан только один дополнительный параметр, то считается, что это `start`, а значение `end` равно длине строки.

Input	Output
`print(is_reading_frame("AGCATGGTAGCTTAGTCAGGTATGATGGGGAT"))`	`True`

P. *Статистика длин слов*

Дана строка, состоящая из слов, разделённых пробелами. Слова составлены из латинских букв. Длина одного слова не превышает 50 символов.
Требуется вывести статистику по длинам слов. Для всех длин слов, которые встречаются в строке надо вывести строку `WORDS OF LENGTH <длина>: <количество слов>`. Одинаковые слова учитываются столько раз, сколько они встречаются. Формат вывода описан в примере.
Указание: создайте массив `len_count` из 51 нуля (51 для того, чтобы в нём был элемент с индексом 50). `len_count[k]` — количество слов длины `k`.

Input	Output

`ehal greka cherez reku vidit greka v reke rak`	`WORDS OF LENGTH 1: 1`
	`WORDS OF LENGTH 3: 1`
	`WORDS OF LENGTH 4: 3`
	`WORDS OF LENGTH 5: 3`
	`WORDS OF LENGTH 6: 1`

Q. *Статистика длин слов — много строк*

Решите задачу «*Статистика длин слов*», если дано несколько строк.
В первой строке на вход программе даётся количество строк N ($1 \leqslant N \leqslant 1000$), затем даётся N строк.
Формат вывода такой же.
Указание: реализуйте функцию `word_stat`, принимающую на вход два параметра: строку, содержащую слова и массив со статистикой. Функция должна изменять переданный массив — учитывать слова из переданной строки.

Input	Output

`2`
`ehal greka cherez reku`
`vidit greka v reke rak`
	`WORDS OF LENGTH 1: 1`
	`WORDS OF LENGTH 3: 1`
	`WORDS OF LENGTH 4: 3`
	`WORDS OF LENGTH 5: 3`
	`WORDS OF LENGTH 6: 1`

R. *Биологические слова — без пробелов*

Дана строка из заглавных латинских букв, разбитая на куски. Извлечь из неё все «биологические» слова. *«Биологическим» словом* будем называть подстроку данной строки, содержащую только символы `A`, `C`, `G`, `T`, и которая не является подстрокой никакого другого биологического слова.

В первой строке на вход программе даётся количество частей строки N ($1 \leqslant N \leqslant 1000$), затем перечисляется N частей строк.

Оформите решение в виде функции, принимающей на вход одну строку (склеенную из данных частей) и возвращающую требуемый массив «биологических» слов.

Input	Output
3 AAXACARGX CTGTGXERWCGWTGWCTG ASDCCACCDCCCC	["AA", "ACA", "G", "CTGTG", "CG", "TG", "CTGA", "CCACC", "CCCC"]

Комментарий к примеру: четвёртое из слов массива, `CTGTG` является биологическим словом в смысле выше данного определения, а строка `CTGT` — нет.

S. *Самое частое число*

Дан целочисленный массив. Не изменяя его и не используя дополнительные массивы, определите, какое число в этом списке встречается чаще всего.

Если таких чисел несколько, выведите любое из них.

Размер массива не превосходит 10^3.

Input	Output
6 1 2 3 2 3 3	3

T. *Много мутаций*

Теперь вам надо отметить много мутаций в строке.

В первой строке входных данных содержится одна ACGT-строка s, во второй строке целое число — количество мутаций N ($0 \leqslant N < 10^4$). Затем в $2N$ строках вводятся k ($0 \leqslant k < len(s)$) — индекс символа данной ACGT-строки, где произошла мутация и символ, на который надо его заменить.

Программа должна вывести получившуюся строку.

Input	Output
AACGTACGGACTAGC 2 4 C 10 G	AACGCACGGAGTAGC

U* *Рамки считывания*

Есть ACGT-строка. Строка может быть достаточно длинная, поэтому её разбили на части. На вход программе даётся количество таких частей N ($1 \leqslant N \leqslant 1000$), затем N ACGT-строк.

Программа должна вывести последовательности нуклеотидов, находящиеся между старт и стоп-кодонами. Эти последовательности, вообще говоря, не обязательно имеют длину, кратную трём.

Можно считать, что рамки считывания не пересекаются (как это и происходит на самом деле).

Input	Output
3 AGCATGGCAGCTTAGTCAGGTATGATTGGGAT GACCTAACGAATGGGATTAGAGTCTCTTTTGG AATAAGCCTGAATGATCCGAGTAACATGACAG	GCAGCT ATTGGGA GGAT ATCCGAG

Комментарий к примеру:

Обратите внимание, что открывающий триплет должен «дождаться» закрывающего.

AGC<u>ATG</u>GCAGCT<u>TAG</u>TCAGGT<u>ATG</u>ATTGGGA<u>TGA</u>CCTAACGA<u>ATG</u>GGAT<u>TAG</u>AGTCTCTTTTGGAATAAGCCTGA<u>ATG</u>ATCCGAG<u>TAA</u>CATGACAG

V* *Статистика по триплетам*

Вам надо определить частоту встречаемости триплетов в данной ACGT-строке. На вход программе даётся одна ACGT-строка, состоящая не менее, чем из 3 символов.

Программа должна вывести столько строк, сколько разных триплетов встречается в данной строке. В каждой такой строке должен быть указан триплет, а после пробела — их количество в данной строке.

Выводимые триплеты должны быть упорядочены в алфавитном порядке.

Эту задачу можно решить заметно короче, используя пока неизученную вами структуру данных. Зато, используя массив, придётся немного разобраться с комбинаторикой и отдать должное Георгию Гамову (статья в Wikipedia), который не только первый заметил, что $64 = 4^3 > 20$, но и понял, что это может значить.

Input	Output
`ACGT`	`ACG 1` `CGT 1`

`ATCGACTATCG`	`ACT 1`
	`ATC 2`
	`CGA 1`
	`CTA 1`
	`GAC 1`
	`TAT 1`
	`TCG 2`