Использование сортировки

A. *Все триплеты по возрастанию*
Дана строка, содержащая одну ACGT-последовательность. Вывести все триплеты этой строки в лексикографическом порядке.

Input	Output
CCCACCC	ACC CAC CCA CCC CCC

B. *Все различные триплеты по возрастанию*
Дана строка, содержащая одну ACGT-последовательность. Вывести все триплеты этой строки в лексикографическом порядке. Каждый триплет должен встречаться не более одного раза.

Input	Output
CCCACCC	ACC CAC CCA CCC

C. *Все подстроки.*
Дана строка, содержащая одну ACGT-последовательность. Вывести все подстроки этой строки в лексикографическом порядке.

Input	Output
ACGT	A AC ACG ACGT C CG CGT G GT T

D. *Все триплеты с указанием частоты.*
Дана строка, содержащая одну ACGT-последовательность. Вывести все триплеты этой строки в лексикографическом порядке. На каждой строке после триплета укажите сколько раз этот триплет встретился в исходной строке.

Input	Output
CCCACCC	ACC 1 CAC 1 CCA 1 CCC 2

E. *Все слова в обратном порядке.*
Дана строка из слов, записанных строчными латинскими буквами через пробел. Вывести строку, в которой записаны те же слова, но в обратном лексикографическом порядке.

Input	Output
sunul greka ruku v reku	v sunul ruku reku greka

F. *Слова, встречающиеся чаще всего.*
Дана строка, в которой записаны слова из латинских букв. Слова разделены пробелом. Вывести слово, встречающееся чаще всего. Если таких слов несколько, вывести их все в лексикографическом порядке, по одному в строке.

Input	Output
ca cb bc ab bc ab	ab bc
t o p	o p t

G. *Все заголовки из FASTA-файла для длинных цепочек.*

Вывести в алфавитном порядке все заголовки из FASTA-файла, соответствующие длинным цепочкам нуклиотидов. Длинными будем считать цепочки, состоящие не менее чем, из N нуклеотидов. Число N записано в первой строчке данного FASTA-файла.

Если длинных нуклеотидных строчек нет, ничего не выводите.

Input	Output
10 >the very first protein AFNQIJRG HJQE JEQ J >second protein FDRG REDGT >third protein ABCDEFGHIJ >forth protein OGHIJKZ	>the very first protein >third protein

H. *Строки с большим относительным GC-содержанием.*

Дан файл, содержащий некоторое количество ACGT-строк. Вывести в лексикографическом порядке все строки (см. формат вывода), относительное GC-содержание которых не более чем на 10% отличается от максимального среди указанных строк.

Input	Output
GCGCGCGCGGCGGGGA GGGGGGGGGGGGGGC GGGGGGGGGGGGGGT GGCC GGAATT CCCCCCCCT	GCGCGCGCGGCGGGGA GGCC GGGGGGGGGGGGGGC GGGGGGGGGGGGGGT

Сортировка по ключу, генераторы

I. Дан массив слов, содержащих строчные латинские буквы. Упорядочить слова по возрастанию с учётом обратного чтения, т.е. не по начальным, а по конечным буквам (упорядоченный таким образом словарь называется *обратным* и удобен при изучении особенностей строения конца слов и вообще словообразования, подборе рифм).

В единственной строке вводится N ($1 \leqslant N \leqslant 10^3$) слов из строчных букв длиной не более 50 символов, разделённых пробелами.

Input	Output
`window table ten screen`	`table screen ten window`

Определение. Сортировка называется *стабильной*, если она не меняет местами равные элементы исходного массива.

При первом прочтении определения может возникнуть впечатление, что оно имеет мало смысла. Действительно, ведь равные элементы неразличимы и нет никакой разницы, в каком порядке они идут. Это совсем не так. Достаточно хорошей иллюстрацией могут служить электронные таблицы (Excel, GoogleSheets, и проч.). Представьте таблицу с двумя столбцами, в первом записаны фамилии учеников, во втором класс, в котором учится каждый из учеников. Мы хотим расположить строки этой таблицы так, чтобы сначала были все ученики первого класса в алфавитном порядке, затем второго, также в алфавитном порядке, и так далее. Для этого нам достаточно сначала отсортировать таблицу по первому столбцу, а затем по второму. Благодаря тому, что в таблицах используется стабильная сортировка, после того, как мы отсортировали учеников по классам, среди равных элементов (учеников из одного класса) исходный порядок (алфавитный) не нарушится.

J. *Благосостояние и стабильность.*

В некой стране есть три разных валюты — динары, лиры и тугрики. Известно, что по текущему курсу валют, одна лира равна пяти динарам, а один тугрик равен семи динарам. Журнал «Благосостояние и стабильность» каждый год публикует список самых состоятельных граждан (не более 1000). Главный казначей страны выдает редактору журнала список главных богачей в следующем формате:
`<фамилия> <количество динаров> <количество лир> <количество тугриков>.`
Список составлен в порядке влиятельности богачей. Ваша задача отсортировать список по общему благосостоянию, причем если двое граждан имеют одинаковое количество денег, в вашем списке они должны быть расположены в порядке влиятельности (сортировка должна быть *стабильной*).

В первой строке задается число N — количество людей в списке казначея ($N \leqslant 1000$) . В последующих N строках задается список казначея в описанном формате. Выведите список фамилий самых богатых людей. Каждую фамилию нужно выводить на отдельной строке.

Input	Output

`4`
`Ivanov 34 0 0`
`Petrov 0 7 0`
`Sidorov 0 0 5`
`Rosenblum 1000 1000 100500`
	`Rosenblum`
	`Petrov`
	`Sidorov`
	`Ivanov`

K. *Результаты олимпиады*

Во время проведения олимпиады каждый из участников получил свой идентификационный номер — целое неотрицательное число. Необходимо отсортировать список участников олимпиады по количеству набранных ими баллов.

На первой строке дано число N ($1 \leqslant N \leqslant 1000$) — количество участников. В следующих N строках даны идентификационный номер и набранное число баллов соответствующего участника. Во входных данных могут встретиться строки с совпадающими идентификационными номерами. Все числа во входном файле не превышают 10^5.

В выходной файл выведите исходный список в порядке убывания баллов. Если у некоторых участников одинаковые баллы, то их между собой нужно упорядочить в порядке возрастания идентификационного номера.

Input	Output
3 101 80 305 90 200 14	305 90 101 80 200 14
3 20 80 30 90 25 90	25 90 30 90 20 80

L. *Анаграммы*

Слово называется анаграммой другого слова, если оно может быть получено перестановкой его букв.

Даны два слова на отдельных строках. Слова состоят из строчных латинских букв и цифр. Длины слов не превышают 10^6.

Требуется вывести `YES` — если введенные слова являются анаграммами друг друга, `NO` — если нет.

Input	Output
sharm marsh	YES
ananas nnaass	NO

M. *Похожие массивы*

Назовём два массива похожими, если они состоят из одних и тех же элементов (без учёта кратности). По двум данным массивам выясните, похожие они или нет.

В первой строке содержится число N ($1 \leqslant N \leqslant 100000$) — размер первого массива. Во второй строке идет N целых чисел, не превосходящих по модулю 10^9 — элементы массива. Далее аналогично задается второй массив.

В решении запрещается использовать вспомогательные массивы.

Программа должна вывести слово `YES`, если массивы похожи, и слово `NO` в противном случае.

Input	Output
3 5 7 5 3 7 5 7	YES
3 5 7 5 4 7 5 7 57	NO

N. *Числа*

Саша и Катя учатся в начальной школе. Для изучения арифметики при этом используются карточки, на которых написаны цифры (на каждой карточке написана ровно одна цифра). Однажды они пришли на урок математики, и Саша, используя все свои карточки, показал число A, а Катя показала число B. Учитель тогда захотел дать им такую задачу, чтобы ответ на нее смогли показать и Саша, и Катя, каждый используя только свои карточки. При этом учитель хочет, чтобы искомое число было максимальным.

Во входном файле записано два целых неотрицательных числа A и B (каждое число в одной строке). Длина каждого из чисел не превосходит 100000 цифр.

Выведите одно число — максимальное целое число, которое можно составить используя как цифры первого числа, так и цифры второго числа. Если же ни одного такого числа составить нельзя, выведите -1.

Обратите внимание на тест, в котором у Саши и Кати нет общих цифр, кроме нуля.

Input	Output
280138 798081	8810
123 456	-1
12300 405006	0
13579 4321	31