Сортировка-1. Реализация.

A. *Сортировка выбором максимума*

Требуется отсортировать массив по неубыванию методом “выбор максимума” (сортировка исключениями).

В первой строке вводится одно натуральное число N, не превосходящее 1000 — размер массива. Во второй строке задаются N чисел — элементы массива (целые числа, не превосходящие по модулю 1000).

Требуется вывести получившийся массив.

Input	Output
2 3 1	1 3

B. *Сортировка вставками*

Требуется отсортировать массив по неубыванию методом «вставок» (сортировка включениями).

В алгоритме сортировки вставкой в каждый произвольный момент начальная часть списка уже отсортирована. В решении имеется цикл `for i in range(1, len(A))`, внутри которого в предположении, что элементы списка `A[0], A[1], ..., A[k-1]` уже отсортированы, элемент `A[k]` добавляется в отсортированную часть списка.

В первой строке вводится одно натуральное число N, не превосходящее 1000 — размер массива. Во второй строке задаются N чисел — элементы массива (целые числа, не превосходящие по модулю 1000).

Требуется вывести получившийся массив.

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

C. *Сортировка вставками – иллюстрация*

Продемонстрируйте работу метода сортировки вставками по возрастанию. Для этого выведите состояние данного массива после каждой вставки на отдельных строках. Если массив упорядочен изначально, то следует не выводить ничего.

На первой строке дано число N ($1 \leqslant N \leqslant 100$) — количество элементов в массиве. На второй строке задан сам массив: последовательность натуральных чисел, не превышающих 10^9.

В выходной файл выведите строки (по количеству вставок) по N чисел каждая.

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

D. *Сортировка пузырьком*

Требуется отсортировать массив по неубыванию методом "пузырька".

В первой строке вводится одно натуральное число N, не превосходящее 1000 — размер массива. Во второй строке задаются N чисел — элементы массива (целые числа, не превосходящие по модулю 1000).

Требуется вывести получившийся массив.

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

E. *Сортировка пузырьком – количество обменов*

Определите, сколько обменов сделает алгоритм пузырьковой сортировки по возрастанию для данного массива.

На первой строке дано число N ($1 \leqslant N \leqslant 1000$) — количество элементов в массиве. На второй строке — сам массив. Гарантируется, что все элементы массива различны и не превышают по модулю 10^9.

Выведите одно число — количество обменов пузырьковой сортировки.

Input	Output
5 1 2 3 4 5	0
5 5 4 3 2 1	10

F. *Поразрядная сортировка – иллюстрация*

Поразрядная сортировка является одним из видов сортировки, которые работают за линейное от размера сортируемого массива время. Такая скорость достигается за счет того, что эта сортировка использует внутреннюю структуру сортируемых объектов. Изначально этот алгоритм использовался для сортировки перфокарт. Первая его компьютерная реализация была создана в университете MIT Гарольдом Сьюардом (Harold H. Seward). Опишем алгоритм подробнее.

Пусть задан массив строк $s_1, \ldots, s_i$ причем все строки имеют одинаковую длину m. Работа алгоритма состоит из m фаз. На i-ой фазе строки сортируются по i-ой с конца букве. Происходит это следующим образом. В этой задаче рассматриваются строки из цифр от 0 до 9. Для каждой цифры создается "корзина" ("bucket"), после чего строки s_i распределяются по "корзинам" в соответствии с i-ой с конца цифрой. Строки, у которых i-ая с конца цифра равна j попадают в j-ую корзину (например, строка 123 на первой фазе попадет в третью корзину, на второй — во вторую, на третьей — в первую). После этого элементы извлекаются из корзин в порядке увеличения номера корзины.

Таким образом, после первой фазы строки отсортированы по последней цифре, после двух фаз — по двум последним,..., после m фаз — по всем. При этом важно, чтобы элементы в корзинах сохраняли тот же порядок, что и в исходном массиве (до начала этой фазы). Например, если массив до первой фазы имеет вид: $111, 112, 211, 311$, то элементы по корзинам распределятся следующим образом: в первой корзине будет $111, 211, 311$, а второй: 112. Ваша задача состоит в написании программы, детально показывающей работу этого алгоритма на заданном массиве.

Первая строка входного файла содержит целое число n ($1 \leqslant n \leqslant 1000$). Последующие n строк содержат каждая по одной строке s_i. Длины всех s_i одинаковы и не превосходят 20. Все s_i состоят только из цифр от 0 до 9.

В выходной файл выведите исходный массив строк в, состояние "корзин" после распределения элементов по ним для каждой фазы и отсортированный массив. Следуйте формату, приведенному в примере.

Input	Output
9 12 32 45 67 98 29 61 35 09	Initial array: 12, 32, 45, 67, 98, 29, 61, 35, 09 ********** Phase 1 Bucket 0: empty Bucket 1: 61 Bucket 2: 12, 32 Bucket 3: empty Bucket 4: empty Bucket 5: 45, 35 Bucket 6: empty Bucket 7: 67 Bucket 8: 98 Bucket 9: 29, 09 ********** Phase 2 Bucket 0: 09 Bucket 1: 12 Bucket 2: 29 Bucket 3: 32, 35 Bucket 4: 45 Bucket 5: empty Bucket 6: 61, 67 Bucket 7: empty Bucket 8: empty Bucket 9: 98 ********** Sorted array: 09, 12, 29, 32, 35, 45, 61, 67, 98

G. *Mergesort*

Отсортируйте данный массив, используя сортировку слиянием. Можете использовать рекурсивную функцию или написать нерекурсивный алгоритм.

В этой задаче рекомендуется реализовать функцию `merge(x, a, b, c)` для слияния двух упорядоченных частей массива, в которой разрешается использовать вспомогательный массив. Здесь `x` - данный массив, который будет изменён в ходе работы функции, `a, b, c` — границы соседних упорядоченных частей массива.

Input	Output
2 3 1	1 3

H. *Quicksort*

Идея быстрой сортировки заключается в выборе некоторого элемента `x` массива `A` и последующем разбиении исходного массива на три части: первая состоит из элементов, мéньших x, вторая — равных `x` и третья — бóльших `x`. Эту часть алгоритма лучше реализовать в виде функции. Затем рекурсивно вызывать функцию сортировки от первой и третьей частей.

Элемент, относительно которого происходит разбиение называется *опорным* (*pivot*). Задача была в листочке про массивы (доп. листок, задача `C`).

Скорость работы алгоритма существенно зависит от метода выбора опорного элемента.

В программе запрещается использовать вспомогательные массивы.

Input	Output
2 3 1	1 3

I. *Сортировка подсчётом*

Имеется список целых чисел размера N, сами числа по модулю не превосходят 10000. Отсортировать их за время $O(N)$. Разрешается использовать дополнительную память, объём которой пропорционален максимально возможному значению элемента массива.

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