Задача 1. Таблица инверсий

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая для заданной перестановки строит соответствующую ей таблицу инверсий, и решить с её помощью тестовую задачу.

Функция должна иметь сигнатуру:

```
void permut_to_invtab (int a[], int b[], int n);
```

Здесь `a` — имя массива, содержащего перестановку, `n` — его длина, а `b` — имя массива, в который нужно записать построенную таблицу инверсий.

Формат входных данных

В первой строке входного файла записано целое число N – длина перестановки $(1 \leqslant N \leqslant 1000)$.

Во второй строке через пробел записаны различные натуральные числа $a_1, a_2, \ldots a_N$, принимающие значения от 1 до N — перестановка.

Формат выходных данных

В выходной файл необходимо вывести через пробел N целых чисел, которые будут образовывать таблицу инверсий для заданной перестановки.

Пример

input.txt	output.txt
8 5 2 7 3 8 6 4 1	7 1 2 4 0 2 0 0

Задача 2. Восстановление перестановки

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая по заданной таблице инверсий восстанавливает соответствующую ей перестановку, и решить с её помощью тестовую задачу.

Функция должна иметь сигнатуру:

```
int invtab_to_permut (int b[], int a[], int n);
```

Здесь `b` — имя массива, содержащего таблицу инверсий, `n` — его длина, `a` — имя массива, в который нужно записать восстановленную перестановку.

Функция возвращает 1, если восстановление прошло корректно, иначе она возвращает 0.

Формат входных данных

В первой строке входного файла записано целое число N – длина таблицы инверсий ($1 \leqslant N \leqslant 10^3$).

Во второй строке через пробел записаны натуральные числа $a_1, a_2, \ldots a_N$, принимающие значения в диапазоне от 0 до $N-1$, образующие таблицу инверсий.

Формат выходных данных

В выходной файл необходимо вывести соответствующую заданной таблице инверсий перестановку.

Если таблица инверсий задана некорректно, то вывести слово `NO`.

Примеры

input.txt	output.txt
8 7 1 2 4 0 2 0 0	5 2 7 3 8 6 4 1
3 1 0 1	NO

Задача 3. Следующая по алфавиту перестановка

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая для заданной перестановки чисел от 1 до N строит следующую за ней по алфавиту перестановку, и решить с её помощью тестовую задачу.

Функция должна иметь сигнатуру:

```
void next_permut ( int a[], int n);
```

Здесь `a` — имя массива, содержащего перестановку, `n` — его длина.

Функция должна записать следующую по алфавиту перестановку в тот же массив, не используя дополнительной памяти. Если исходная перестановка — последняя по алфавиту, то следующей для нее будет первая по алфавиту.

Формат входных данных

В первой строке входного файла записано целое число N – длина перестановки ($1 \leqslant N \leqslant 10^3$).

Во второй строке через пробел записаны `различные` натуральные числа $a_1, a_2, \ldots a_N$, принимающие значения от 1 до N.

Формат выходных данных

В выходной файл необходимо вывести через пробел N заданных чисел, которые будут образовывать следующую по алфавиту перестановку для заданной.

Пример

input.txt	output.txt
8 5 2 7 3 8 6 4 1	5 2 7 4 1 3 6 8

Задача 4. Буквы алфавита

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Вам дана строка, состоящая из строчных букв латинского алфавита. Все буквы в ней различны.

Требуется переставить буквы данной строки так, чтобы получившаяся строка была лексикографически больше исходной.

Поскольку в данной постановке задача слишком простая и может иметь не единственное решение, то вам требуется среди всех таких строк выбрать лексикографически минимальную.

Строка s, состоящая из символов s_0, s_1, …, s_n, считается *лексикографически меньше* строки t, состоящей из символов t_0, t_1, …, t_n, если сущесвует индекс k такой, что:

- $s_i = t_i$ для всех $i = 0, 1, \ldots, k-1$;
- $s_k < t_k$.

Иными словами, лексикографическое сравнение строк — это привычное нам сравнение слов «по алфавиту», когда мы находим первую букву, в которой две строки различаются, и на основании этой буквы делалем вывод о том, какое из слов «меньше». Лексикографическое сравнение окружает нас повсюду: его можно найти в порядке людей в списках групп, в порядке номеров в телефонной книге, и т.д.

Формат входных данных

В первой строке входного файла записано целое число N — количество символов в строке ($2 \leqslant N \leqslant 26$).

Во второй строке через пробел записано N строчных букв латинского алфавита. Гарантируется, что все буквы различны.

Формат выходных данных

В выходной файл необходимо вывести через пробел символы требуемой строки.

Гарантируется, что требуемая перестановка существует.

Примеры

input.txt	output.txt
5 a b c d e	a b c e d
3 q z w	w q z