Задача A. Мутанты

Имя входного файла:	`mutants.in`
Имя выходного файла:	`mutants.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	128 мегабайта

Уже долгое время в Институте Искусств, Мутантов и Информационных Технологий разводят милых разноцветных зверюшек. Для удобства каждый цвет обозначен своим номером, всего цветов не более 10^9. В один из прекрасных дней в питомнике случилось чудо: все зверюшки выстроились в ряд в порядке возрастания цветов. Пользуясь случаем, лаборанты решили посчитать, сколько зверюшек разных цветов живет в питомнике, и, по закону жанра, попросили вас написать программу, которая поможет им в решении этой нелегкой задачи.

Формат входного файла

В первой строке входного файла содержится единственное число N ($0 \leqslant N \leqslant 10^5$) — количество зверюшек в Институте. В следующей строке находятся N упорядоченных по неубыванию неотрицательных целых чисел, не превосходящих 10^9 и разделенных пробелами — их цвета. В третьей строке файла записано число M ($1 \leqslant M \leqslant 100\,000$) — количество запросов вашей программе, в следующей строке через пробел записаны M целых неотрицательных чисел (не превышающих $10^9 + 1$).

Формат выходного файла

Выходной файл должен содержать M строчек. Для каждого запроса выведите число зверюшек заданного цвета в питомнике.

Примеры

mutants.in	mutants.out
10 1 1 3 3 5 7 9 18 18 57 5 57 3 9 1 179	1 2 1 2 0

Задача B. Корень кубического уравнения

Имя входного файла:	`cubroot.in`
Имя выходного файла:	`cubroot.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дано кубическое уравнение $ax^3+bx^2+cx+d = 0$ ($a \neq 0$). Известно, что у этого уравнения есть ровно один корень. Требуется его найти.

Формат входного файла

Во входном файле через пробел записаны четыре целых числа: $-1000 \leqslant a, b, c, d \leqslant 1000$.

Формат выходного файла

Выведите единственный корень уравнения с точностью не менее 6 знаков после десятичной точки.

Примеры

cubroot.in	cubroot.out
1 -3 3 -1	1.0000003749
-1 -6 -12 -7	-1.0000000111

Задача C. Мороженое

Имя входного файла:	`ice-cream.in`
Имя выходного файла:	`ice-cream.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вдоль моря узкой полоской тянется пляж. В некоторых точках пляжа расположены ларьки с мороженым. В один прекрасный день не все мороженщики вышли на работу. Распределите мороженщиков по ларькам так, чтобы минимальное расстояние между мороженщиками было как можно больше. Так они будут меньше мешать друг другу.

Формат входного файла

В первой строке вводятся числа N ($2 < N < 10\,001$) — количество ларьков и K ($1 < K < N$) — количество мороженщиков, вышедших на работу. Во второй строке задаются N натуральных чисел в порядке возрастания — координаты ларьков (координаты не превосходят 10^9).

Формат выходного файла

Выведите одно число — минимальное расстояние между соседними ларьками в оптимальной расстановке.

Примеры

ice-cream.in	ice-cream.out
5 3 1 2 3 100 1000	99

Задача D. Быстрая сортировка

Имя входного файла:	`sort.in`
Имя выходного файла:	`sort.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью алгоритма быстрой сортировки (qsort).

Формат входного файла

В первой строке входного файла содержится число N ($1 \le N \le 100\,000$) — количество элементов в массиве. Во второй строке находятся N целых чисел, по модулю не превосходящих 10^9.

Формат выходного файла

В выходной файл надо вывести этот же массив в порядке неубывания, между любыми двумя числами должен стоять ровно один пробел.

Примеры

sort.in	sort.out
10 1 8 2 1 4 7 3 2 3 6	1 1 2 2 3 3 4 6 7 8

Note

В этой задаче обязательно использовать быструю сортировку.

Задача E. Двоичное дерево

Имя входного файла:	`binary.in`
Имя выходного файла:	`binary.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

В исходном файле находится описание двоичного дерева поиска, все элементы которого различны и являются натуральными числами. Дерево построено таким образом, что элементы, большие корня, находятся в правом поддереве, а меньшие — в левом (это свойство верно также для любого поддерева). Известно также, что дерево является сбалансированным и все его уровни полностью заполнены. Также есть M запросов в дереве. Напишите программу, которая для каждого запроса определяет значение родителя, левого и правого сыновей.

Формат входного файла

В первой строке файла записано число N (натуральное, не превышает 20) — количество уровней дерева, во второй строке — элементы дерева в порядке их расположения на уровнях: сначала все элементы первого уровня слева направо, затем второго уровня и т.д. В третьей строке указано число M – количество запросов (натуральное, не превышает 10^4). В четвертой строке записано M натуральных чисел, не превышающих $2 * 10^9$ — запросы к программе.

Гарантируется, что запрашиваемые числа существуют в дереве.

Формат выходного файла

Для каждого запроса требуется вывести в выходной файл в отдельной строке три числа: значения родителя, левого и правого сыновей. В случае, если один из них отсутствует, вместо него вывести -1.

Примеры

binary.in	binary.out
2 5 2 7 2 5 2	-1 2 7 5 -1 -1
3 15 10 50 5 12 36 74 4 10 15 50 12	15 5 12 -1 10 50 15 36 74 10 -1 -1