

## Задача 1. Кастинг

Имя входного файла: `casting.in`  
Имя выходного файла: `casting.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

В театре работает  $n$  актеров. Известно, что среди них  $a$  – высоких,  $b$  – голубоглазых и  $c$  – блондинов. Для главной роли в новом спектакле режиссеру требуется только один высокий голубоглазый блондин. Чтобы спланировать свое время для беседы с каждым таким артистом из труппы театра, режиссеру необходимо узнать, какое максимальное или какое минимальное количество артистов из работающих в театре подходит для этой роли.

Требуется написать программу, которая по заданным числам  $n$ ,  $a$ ,  $b$  и  $c$  определяет минимальное или максимальное количество актеров, с которыми режиссер должен переговорить.

### Формат входного файла

Первая строка входного файла содержит одно число, которое задает, минимальное или максимальное количество актеров необходимо найти в данном тесте. Это число может принимать следующие значения:

- 1, если в данном тесте требуется определить минимальное количество актеров;
- 2, если в данном тесте требуется определить максимальное количество актеров.

Вторая строка входного файла содержит разделенные пробелами четыре целых числа:  $n$ ,  $a$ ,  $b$ ,  $c$  ( $1 \leq n \leq 10\,000$ ,  $0 \leq a \leq n$ ,  $0 \leq b \leq n$ ,  $0 \leq c \leq n$ ).

### Формат выходного файла

Выходной файл должен содержать одно число – минимальное или максимальное (в зависимости от входных данных) количество актеров, которые могут претендовать на главную роль в новом спектакле.

### Примеры входных и выходных файлов

<code>casting.in</code>	<code>casting.out</code>
2 5 3 4 5	3
1 5 3 4 5	2

### Пояснения к примерам

В первом примере, поскольку высоких актеров всего трое, то на главную роль не может подойти больше трех человек.

Во втором примере все актеры – блондины и все, кроме одного, – голубоглазые. Тогда среди трех высоких актеров найдутся хотя бы два голубоглазых (и, естественно, они будут блондинами). Следовательно, минимум два актера точно подойдут на главную роль в новом спектакле.

### Система оценивания

Правильные решения для тестов, в которых требуется найти минимальное количество актеров, будут оцениваться из 50 баллов.

Правильные решения для тестов, в которых требуется найти максимальное количество актеров, будут оцениваться из 50 баллов.

Несмотря на выделение отдельных групп тестов для минимального и максимального количества артистов, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов из условия задачи.

## Задача 2. Города

Имя входного файла: `cities.in`  
Имя выходного файла: `cities.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Юный программист решил придумать собственную игру. Игра происходит на поле размером  $N \times N$  клеток, в некоторых клетках которого расположены города (каждый город занимает одну клетку; в каждой клетке может располагаться не более одного города). Всего должно быть **чётное** количество городов.

Изначально про каждую клетку игрового поля известно, расположен ли в ней город или нет. Чтобы начать игру, необходимо разделить игровое поле на два государства так, чтобы в каждом государстве было **поровну** клеток-городов.

Граница между государствами должна проходить по границам клеток таким образом, чтобы из любой клетки каждого государства существовал путь по клеткам этого же государства в любую другую его клетку (из клетки можно перейти в соседнюю, если они имеют общую сторону). Каждая клетка игрового поля должна принадлежать только одному из двух государств, при этом государства не обязаны состоять из одинакового количества клеток.

Требуется написать программу, которая с учетом сказанного разделит клетки заданного игрового поля между двумя государствами.

### Формат входного файла

Первая строка входного файла содержит одно целое положительное число  $N$ , задающее размер игрового поля ( $1 \leq N \leq 50$ ).

Последующие  $N$  строк содержат по  $N$  заглавных латинских букв (без пробелов), кодирующих соответствующие клетки игрового поля: 'C' обозначает клетку, занятую городом, 'D' – пустую клетку. Гарантируется, что на поле есть хотя бы два города и всего их четное число.

### Формат выходного файла

Выходной файл должен содержать  $N$  строк по  $N$  цифр (без пробелов) в каждой, кодирующих соответствующие клетки. Цифра 1 обозначает, что данная клетка принадлежит первому государству, цифра 2 – данная клетка принадлежит второму государству.

Если решений несколько, необходимо вывести любое из них.

### Примеры входных и выходных файлов

<code>cities.in</code>	<code>cities.out</code>
3	222
DDD	212
DDC	211
DDC	
5	11111
DDDDD	12221
CDCDC	12221
DCCDC	11111
DDDDD	11111
DDDDD	

### ***Система оценивания***

Правильные решения для тестов, в которых всего два города, будут оцениваться из 40 баллов.

Несмотря на выделение отдельной группы тестов с двумя городами, на окончательную проверку будут приниматься только решения, правильно работающие также для всех тестов из условия задачи.

### Задача 3. $A + B = C$

Имя входного файла: `aplusb.in`  
Имя выходного файла: `aplusb.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Часто для пробного тура на различных олимпиадах по информатике предлагается задача « $A + B$ », в которой по заданным целым числам  $A$  и  $B$  требуется найти их сумму.

При проведении городской олимпиады по информатике председатель жюри решил сам подготовить тесты для такой задачи. Для этого он использовал свою оригинальную методику, которая заключалась в следующем: сначала готовятся предполагаемые правильные ответы, а затем подбираются входные данные, соответствующие этим ответам.

Пусть председатель жюри выбрал число  $C$ , запись которого состоит из  $n$  десятичных цифр и не начинается с нуля. Теперь он хочет подобрать такие целые положительные числа  $A$  и  $B$ , чтобы их сумма была равна  $C$ , и запись каждого из них также состояла из  $n$  десятичных цифр и не начиналась с нуля. В дополнение к этому председатель жюри старается подобрать такие числа  $A$  и  $B$ , чтобы каждое из них было *красивым*. Красивым в его понимании является число, запись которого не содержит двух одинаковых подряд идущих цифр. Например, число 1272 считается красивым, а число 1227 — нет.

Требуется написать программу, которая для заданного натурального числа  $C$  вычисляет количество пар красивых положительных чисел  $A$  и  $B$ , сумма которых равна  $C$ . Поскольку количество пар красивых чисел может быть большим, необходимо вывести остаток от деления этого количества на число  $10^9+7$ .

#### Формат входного файла

Входной файл содержит одно целое положительное число  $C$ . Число  $C$  не начинается с нуля. Количество цифр в записи числа  $C$  не превышает 10 000.

#### Формат выходного файла

Выходной файл должен содержать одно целое число — остаток от деления количества искомых пар красивых чисел  $A$  и  $B$  на число  $10^9+7$ .

#### Примеры входных и выходных файлов

<code>aplusb.in</code>	<code>aplusb.out</code>
22	2
200	0
1000	0
239	16

#### Пояснения к примерам

Число 22 можно представить в виде суммы двузначных чисел тремя способами:  $10 + 12$ ,  $11 + 11$ ,  $12 + 10$ . Способ  $11 + 11$  не подходит, поскольку число 11 не является красивым. Следовательно, ответ для числа 22 равен 2.

Число 200 можно представить в виде суммы трехзначных чисел единственным способом:  $100 + 100$ . Этот способ не подходит, поэтому ответ для числа 200 равен 0.

Число 1000 нельзя представить в виде суммы четырехзначных чисел, поэтому ответ для числа 1000 аналогично равен 0.

### **Система оценивания**

Правильные решения для тестов, в которых  $1 \leq C \leq 999$  ( $1 \leq n \leq 3$ ), будут оцениваться из 25 баллов.

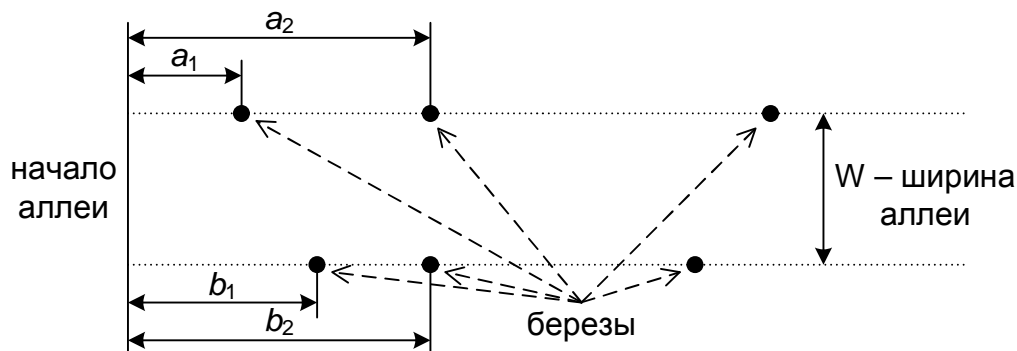
Правильные решения для тестов, в которых  $1 \leq C \leq 999\,999$  ( $1 \leq n \leq 6$ ), будут оцениваться из 50 баллов.

Несмотря на выделение отдельных групп тестов для различных длин числа  $C$ , на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов из условия задачи.

## Задача 4. Березовая аллея

Имя входного файла: birch.in  
Имя выходного файла: birch.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

На краю деревни растет старая березовая аллея. Аллея имеет форму прямой полосы шириной  $W$  метров. Вдоль левой стороны аллеи растет  $N$  берез, а вдоль правой —  $M$  берез, при этом  $i$ -я береза с левой стороны аллеи находится на расстоянии  $a_i$  метров от начала аллеи, а  $j$ -я береза с правой стороны — на расстоянии  $b_j$  метров от начала аллеи.



Отдыхая в деревне прошедшим летом, один юный информатик обнаружил, что кору берез стали грызть зайцы. Чтобы защитить деревья от зайцев, мальчик решил окружить березы красной лентой (зайцы не любят красный цвет и не станут заходить на огражденную лентой территорию. К сожалению, в его распоряжении оказалась только лента длиной  $L$  метров, которую, к тому же, нельзя было разрезать. Единственное, что можно было делать в этом случае — окружить этой лентой как можно больше берез. При этом, чтобы сохранить аллею, необходимо окружить на каждой стороне аллеи хотя бы одну березу.

Требуется написать программу, которая по заданной длине ленты, ширине аллеи и положению берез на ней определяет максимальное число берез, которое можно окружить этой лентой. Считается, что березы представляются точками, толщиной берез и шириной ленты следует пренебречь.

### Формат входного файла

Первая строка входного файла содержит два разделенных пробелом целых числа: длину ленты  $L$  и ширину аллеи  $W$  ( $1 \leq L \leq 2 \times 10^5$ ,  $1 \leq W \leq 10^4$ ).

Вторая и третья строки описывают березы вдоль левой стороны аллеи. Вторая строка содержит число  $N$  — количество берез ( $1 \leq N \leq 2000$ ), а третья строка содержит  $N$  различных целых чисел  $a_1, a_2, \dots, a_N$  ( $0 \leq a_i \leq 10^5$ ), заданных по возрастанию.

Четвертая и пятая строки описывают березы вдоль правой стороны аллеи. Четвертая строка содержит число  $M$  — количество берез ( $1 \leq M \leq 2000$ ), а пятая строка содержит  $M$  различных целых чисел  $b_1, b_2, \dots, b_M$  ( $0 \leq b_i \leq 10^5$ ), заданных по возрастанию.

### Формат выходного файла

Выходной файл должен содержать одно целое число: максимальное количество берез, которое можно оградить заданной лентой.

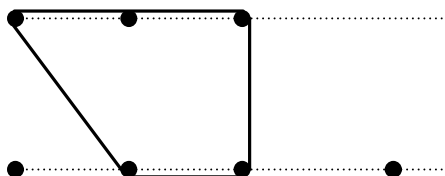
Гарантируется, что если максимальное число берез, которое можно оградить лентой длины  $L$ , равно  $X$ , то нет способа оградить  $(X + 1)$  березу лентой длины  $(L + 10^{-5})$ .

### Примеры входных и выходных файлов

<code>birch.in</code>	<code>birch.out</code>
18 4 3 0 3 6 4 0 3 6 10	5
5 3 1 0 1 0	0

### Пояснения к примерам

В первом примере можно, например, огрaдить березы способом, указанным на рисунке ниже.



Во втором примере длины ленты недостаточно чтобы огрaдить хотя бы по одной березе с каждой стороны.

### Система оценивания

Правильные решения для тестов, в которых  $1 \leq N + M \leq 50$ , будут оцениваться из 30 баллов.

Правильные решения для тестов, в которых  $1 \leq N + M \leq 500$ , будут оцениваться из 60 баллов.