Общая информация по задачам первого тура

Доступ к результатам проверки решений задач во время тура

В течение тура можно не более 10 раз по каждой задаче запросить информацию о результатах оценивания решения на тестах жюри. Запрос по каждой задаче можно делать не чаще одного раза в 5 минут.

Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

Сводная таблица ограничений

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
1. Хоккей на Урале	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
2. Робот-сборщик	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
3. Графический редактор «Хамелеон»	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
4. Древние династии	2 секунды	128 МБ	Для каждой подзадачи сообщаются только баллы за эту подзадачу

Задача 1. Хоккей на Урале

Имя входного файла:	`tournament.in`
Имя выходного файла:	`tournament.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Для популяризации хоккея и повышения мастерства хоккейных команд Урала был организован Всеуральский турнир. Для участия в турнире были приглашены N хоккейных команд из городов Урала.

После первых двух туров, в каждом из которых каждая команда провела по одному матчу, оказалось, что команд слишком много. Организаторами турнира было решено допустить к дальнейшему участию только K команд, никакие две из которых не встречались в рамках первых двух туров.

Требуется написать программу, которая находит набор из K команд, удовлетворяющий условиям, либо выводит сообщение о том, что это сделать невозможно. В случае существования нескольких подходящих наборов необходимо найти любой из них.

Формат входных данных

В первой строке входного файла содержится число N ($2 \leqslant N \leqslant 100\,000$, N — чётное).

Последующие N строк содержат описания всех прошедших матчей. Описание каждого матча состоит из двух натуральных чисел, не превышающих N — номеров команд, игравших в матче. Первые $N/2$ из них соответствуют матчам первого тура, оставшиеся — матчам второго тура.

Последняя строка входного файла содержит одно число K ($2 \leqslant K \leqslant N$).

Гарантируется, что каждая команда сыграла ровно два матча: один в первом туре и один — во втором.

Формат выходных данных

Выходной файл должен содержать либо единственное число 0, если решения не существует, либо K различных чисел — номера отобранных команд.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

Подзадача 1

$N \leqslant 10$. Подзадача оценивается в 30 баллов.

Подзадача 2

$N \leqslant 1000$. Подзадача оценивается в 30 баллов.

Подзадача 3

$N \leqslant 100\,000$. Подзадача оценивается в 40 баллов.

Примеры

tournament.in	tournament.out
6 1 2 3 5 4 6 2 3 4 5 1 6 3	1 4 3
4 1 2 3 4 2 1 4 3 3	0

Задача 2. Робот-сборщик

Имя входного файла:	`robot.in`
Имя выходного файла:	`robot.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Студенты одного из вузов спроектировали робота для частичной автоматизации процесса сборки авиационного двигателя.

В процессе сборки двигателя могут встречаться *операции* 26 типов, которые обозначаются строчными буквами латинского алфавита. Процесс сборки состоит из N операций. Предполагается использовать робота один раз для выполнения части подряд идущих операций из процесса сборки.

Память робота состоит из K ячеек, каждая из которых содержит одну операцию. Операции выполняются последовательно, начиная с первой, в том порядке, в котором они расположены в памяти. Выполнив последнюю из них, робот продолжает работу с первой операции. Робота можно остановить после выполнения любого количества операций. Использование робота *экономически целесообразно*, если он выполнит хотя бы $(K + 1)$ операций.

Требуется написать программу, которая по заданному процессу сборки определяет количество экономически целесообразных способов использования робота.

Формат входных данных

В первой строке входного файла записано число K $(1 \leqslant K < N)$ — количество операций, которые можно записать в память робота.

Вторая строка состоит из N $(2 \leqslant N \leqslant 200\,000)$ строчных латинских букв, обозначающих операции — процесс сборки двигателя. Операции одного и того же типа обозначаются одной и той же буквой.

Формат выходных данных

Выходной файл должен содержать единственное целое число — количество экономически целесообразных способов использования робота.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из соответствующей группы пройдены.

Подзадача 1

$N \leqslant 100$. Подзадача оценивается в 30 баллов.

Подзадача 2

$N \leqslant 5000$. Подзадача оценивается в 30 баллов.

Подзадача 3

$N \leqslant 200\,000$. Подзадача оценивается в 40 баллов.

Примеры

robot.in	robot.out
2 zabacabab	5
2 abc	0

Пояснение к примерам

В первом примере экономически целесообразно использовать робота для выполнения следующих последовательностей операций:

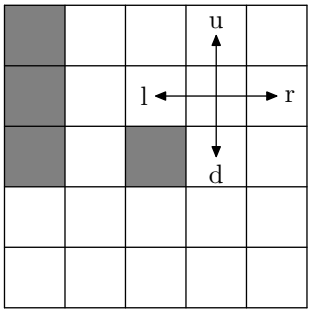
- со 2-й по 4-ю: «aba», при этом в памяти робота содержатся операции «ab»;
- с 4-й по 6-ю: «aca», в памяти робота «ac»;
- с 6-й по 8-ю: «aba», в памяти робота «ab»;
- с 6-й по 9-ю: «abab», в памяти робота «ab»;
- с 7-й по 9-ю: «bab», в памяти робота «ba».

Задача 3. Графический редактор «Хамелеон»

Имя входного файла:	`chameleon.in`
Имя выходного файла:	`chameleon.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Юный информатик осваивает новый графический редактор «Хамелеон». Этот редактор обладает необыкновенной простотой. Он поддерживает ровно два цвета — чёрный и белый, и один инструмент — «Хамелеон».

Поле редактора — это квадрат $N \times N$ клеток. На одной из клеток поля находится курсор-хамелеон. Его можно передвигать в пределах поля в четырех направлениях — вверх, вниз, вправо или влево ровно на одну клетку. Цвет курсора всегда должен совпадать с цветом клетки, в которой он находится. Для этого, когда он перемещается на клетку другого цвета, должно произойти одно из двух событий: либо курсор меняет свой цвет на цвет этой клетки, либо наоборот — клетка меняет свой цвет на цвет курсора. Например, если курсор перемещается из чёрной клетки в белую, либо он должен перекраситься в белый цвет, либо белая клетка, в которой он теперь находится, должна стать чёрной. Если клетка и курсор имеют одинаковый цвет, то их цвет не изменяется.



Изначально курсор имеет чёрный цвет и находится в левой верхней клетке поля. Эта клетка также окрашена в чёрный цвет. Все остальные клетки поля окрашены в белый цвет.

Требуется написать программу, определяющую последовательность действий курсора-хамелеона, после выполнения которой на поле получится картинка, заданная во входных данных.

Формат входных данных

В первой строке входного файла задано число N ($5 \leqslant N \leqslant 100$) — размер поля.

В следующих N строках описывается картинка, которую необходимо получить. Каждая строка описания картинки имеет длину N и состоит из символов «`W`», если соответствующая клетка белая, и «`B`», если чёрная.

Последняя строка файла содержит номер теста.

Формат выходных данных

Выходной файл должен содержать одну строку с описанием искомой последовательности действий.

Для обозначения перемещения влево, вверх, вправо или вниз с изменением **цвета курсора** следует использовать буквы «`l`», «`u`», «`r`» или «`d`» соответственно. Для обозначения перемещения влево, вверх, вправо или вниз с изменением **цвета клетки** следует использовать буквы «`L`», «`U`», «`R`» или «`D`» соответственно. Если курсор перемещается на клетку своего цвета, можно использовать как заглавную, так и строчную букву.

Примечание

В этой задаче тестовые данные доступны участникам олимпиады. Они находятся на вашей рабочей станции в каталоге «`C:\work\chameleon-tests`». Изменить файлы в этом каталоге невозможно. При необходимости изменения файлов можно скопировать их в другой каталог.

Тесты нумеруются в соответствии с названиями файлов от 0 до 20. Тест из примера имеет номер 0, он используется для предварительной проверки. Тесты с номерами с 1 по 20 включительно используются для окончательной проверки.

Система оценивания

Окончательная проверка данной задачи осуществляется на наборе из 20 тестов. Каждый тест оценивается из 5 баллов. Тесты оцениваются независимо.

Тест считается пройденным, если выведенная последовательность содержит не более 5 000 000 действий и приводит к правильному результату.

Первые 10 тестов оцениваются в 5 баллов, если тест пройден.

Оставшиеся 10 тестов оцениваются следующим образом. Если тест пройден, то:

- в 5 баллов, если ответ содержит не более $3N^2$ действий;
- в 4 балла, если ответ содержит не более $5N^2$ действий;
- в 3 балла, если ответ содержит не более $10N^2$ действий;
- в 2 балла, если ответ содержит не более $2{,}5N^3$ действий;
- в 1 балл, если ответ содержит не более 5 000 000 действий.

Пример

chameleon.in	chameleon.out
5 BWWWW BWWWW BWBWW WWWWW WWWWW 0	DDRRdlU

Описание визуализатора

Для просмотра последовательности действий участнику предоставляется визуализатор, запускаемый с помощью файла «`C:\work\chameleon-tests\visualize.cmd`».

При запуске визуализатора без параметров будет предложено выбрать входной и выходной файлы для визуализации. Имя входного и выходного файла также можно указать в виде параметров командной строки, запустив команду «`C:\work\chameleon-tests\visualize.cmd <входной файл> <выходной файл>`».

Задача 4. Древние династии

Имя входного файла:	`dynasties.in`
Имя выходного файла:	`dynasties.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	128 МБ

История Татаро-монгольского ханства богата на правителей. Каждый из N правителей принадлежал к одной из двух династий, причём власть часто переходила от одной династии к другой. Каждое восхождение правителя на престол отмечалось праздником, проводимым 26 марта. В летописях зафиксированы годы проведения этих праздников, причем известно, что правители первой династии устраивали для народа праздник кумыса, а второй — праздник мёда.

На конференции по истории Татаро-монгольского ханства каждый из S учёных предложил свою версию толкования летописи. А именно, i-й историк утверждал, что от каждого праздника кумыса до следующего праздника кумыса проходило не менее KL_i лет, но не более KR_i лет, в то время как от каждого праздника мёда до следующего праздника мёда проходило не менее ML_i лет, но не более MR_i лет.

Каждой предложенной версии может соответствовать несколько распределений правителей по династиям. Ученые договорились считать *показателем сомнительности* распределения число переходов власти к представителю той же самой династии.

Требуется написать программу, которая найдёт распределение, соответствующее хотя бы одной из версий и имеющее наименьший показатель сомнительности, а также версию, которой оно соответствует.

Формат входных данных

В первой строке входного файла записано число N ($2 \leqslant N \leqslant 200\,000$) — количество праздников в летописи. Следующая строка содержит целые числа $X_1, X_2, \ldots, X_N$ ($1 \leqslant X_1 < X_2 < \ldots < X_N \leqslant 10^9$) — годы проведения праздников.

В третьей строке записано число учёных S ($1 \leqslant S \leqslant 50$). В каждой из последующих S строк записаны четыре натуральных числа KL_i, KR_i, ML_i, MR_i ($1 \leqslant KL_i \leqslant KR_i \leqslant 10^9$), ($1 \leqslant ML_i \leqslant MR_i \leqslant 10^9$).

Формат выходных данных

Первая строка выходного файла должна содержать числа P и Q, где P — номер учёного, версии которого соответствует распределение с наименьшим показателем сомнительности, а Q — показатель сомнительности этого распределения.

Вторая строка должна состоять из N цифр 1 и 2, записанных без пробелов, означающих приход к власти представителя первой или второй династии соответственно. Если существует несколько решений с наименьшим показателем сомнительности Q, выведите любое из них.

В случае, если ни в одной из версий учёных не существует способа распределения периодов правления между династиями так, чтобы не нарушались ограничения на промежутки времени между праздниками, выходной файл должен содержать единственное число 0.

Система оценивания

Данная задача содержит пять подзадач. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

Подзадача 1

$2 \leqslant N \leqslant 15$, $1 \leqslant S \leqslant 10$. Подзадача оценивается в 20 баллов.

Подзадача 2

$2 \leqslant N \leqslant 2000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 2000$. Подзадача оценивается в 20 баллов.

Подзадача 3

$2 \leqslant N \leqslant 10\,000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 10\,000$. Подзадача оценивается в 20 баллов.

Подзадача 4

$2 \leqslant N \leqslant 200\,000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 200\,000$. Подзадача оценивается в 20 баллов.

Подзадача 5

$2 \leqslant N \leqslant 200\,000$, $1 \leqslant S \leqslant 50$. Подзадача оценивается в 20 баллов.

Примеры

dynasties.in	dynasties.out
3 1 2 3 1 1 1 1 1	1 1 122
4 1 6 9 13 2 1 2 2 3 6 7 3 3	0
5 3 6 8 9 10 2 2 3 1 1 1 4 1 10	2 0 21212