Вузовско-академическая олимпиада по информатике

2019-2020 учебный год

Задания первого (отборочного) этапа

Задача A. Дом

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Джек хочет построить дом. Будучи хорошим инженером, Джек выяснил, что для постройки дома требуется потратить A минут на планирование дома и B минут на реализацию плана (т.к. Джек — удивительный человек, время у него может быть и отрицательным). И, конечно, ему хочется узнать, сколько минут ему нужно потратить на воплощение своего искусства — дома в жизнь. Сейчас Джек очень занят (быть может, он помогает кому-то разобраться с машиной или моет полы, а может просто пошёл на пикник). Поэтому он просит Вас помочь ему с расчётами.

Формат входных данных

В первой строке дано целое число A, во второй строке — целое число B — время, необходимое на планирование дома и время, необходимое на реализацию плана соответственно. ($-10^{18} \leqslant A, B \leqslant 10^{18}$)

Формат выходных данных

В единственной строке выведите одно целое число — суммарное время, необходимое на постройку дома.

Система оценки

Подзадача 1 (баллы: 10)

$0 \leqslant A, B \leqslant 9$

Подзадача 2 (баллы: 20)

$0 \leqslant A, B \leqslant 10^9$

Подзадача 3 (баллы: 50)

$-10^9 \leqslant A, B \leqslant 10^9$

Подзадача 4 (баллы: 20)

$-10^{18} \leqslant A, B \leqslant 10^{18}$

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Примеры

стандартный ввод	стандартный вывод
1 2	3
5 6	11

Замечание

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача B. Мифы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Недавно по всему миру прокатилась волна восторга всех ученых от невероятного открытия! Знаменитый физик Саша Вилкин доказал теорему, которая мучила великие умы человечества последние 2020 лет!

Эта теорема очень важна для всех отраслей науки, ведь она приближает нас к пониманию того, как же устроена вселенная. Она называется теоремой о N слонах и M черепахах. Всё доказательство занимает более 50 страниц формата A4, но вот некоторые выдержки из научного документа:

1. «Земля стоит на N слонах». Доказательство для произвольного N очень сложное, но чтобы понять, откуда вообще взялось такое утверждение, приведём пример. Например, для $N = 3$ слоны олицетворяют единство, дух и судьбу!

2. «Под каждой ногой каждого слона находится по киту». Тут всё тривиально: подсознательно идёт отсылка к великому философскому тезису о том, что любой порядок зиждется на хаосе.

3. «Каждый кит стоит на M черепахах». Эта часть теоремы до сих пор остаётся непостижимой для множества учёных по всему миру, ведь там затрагиваются практически все области высшей физики, поэтому кратко объяснить, почему это так, у нас не получится.

Гениальной статье Саши Вилкина не хватает только математических расчётов. Как жаль, что Саша Вилкин физик, а не математик! Именно поэтому он надеется на то, что вы сможете ему помочь с невероятно сложной задачей — посчитать, сколько же ног держат нашу Землю. Слоны и черепахи в теореме по своему подобию схожи со своими земными сородичами и имеют ровно 4 ноги, а киты не имеют их вовсе.

Формат входных данных

В первой строке вводится число N — количество слонов, на которых держится Земля.

Во второй строке вводится число M — количество черепахах, на которых стоит один кит.

Все числа во входных данных целые положительные.

Формат выходных данных

Выведите единственное число — ответ на вопрос «Сколько ног держат землю?».

Система оценки

Подзадача 1 (баллы: 55)

$N = 1; 1 \leqslant M \leqslant 1000$

Подзадача 2 (баллы: 30)

$1 \leqslant N, M < 10^9$. Гарантируется, что ответ не превосходит 10^9

Подзадача 3 (баллы: 15)

$1 \leqslant N, M < 10^{18}$. Гарантируется, что ответ не превосходит 10^{18}

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Примеры

стандартный ввод	стандартный вывод
1 1	20
1 2	36
2 1	40
4 21	1360

Замечание

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача C. Национальные интернет ресурсы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Программист 3-го разряда Пётр Иванович пишет Интернет-Обозреватель для принципиально новой Российской Операционной Системы (сокращённо РосОС) в рамках национальной программы «Цифровая экономика». Ну и, конечно же, в национальной операционной системе пользователи должны видеть только национальные интернет-ресурсы! Министры НацМинСвязи считают, что любая строка, которая начинается с «http://» или «https://» и заканчивается на «.ru», является национальным интернет-ресурсом. Пётр Иванович хоть и программист, но всё-таки только 3-го разряда, поэтому вы должны помочь ему научиться определять национальные-интернет ресурсы!

Формат входных данных

Вводится единственная непустая строка — проверяемый адрес. Строка состоит из строчных латинских букв и символов «:», «/», «.». Длина строки не превосходит 127 символов.

Формат выходных данных

Выведите «yes», если адрес является национальным интернет-ресурсом, и «no» в противном случае.

Система оценки

Подзадача 1 (баллы: 50)

Гарантируется, что проверяемый адрес не начинается с «https://».

Для этой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Подзадача 2 (баллы: 50)

Проверяемым адресом может быть любой адрес, соответствующий формату ввода.

Для этой подзадачи необходимо, чтобы ваше решение корректно работало для предыдущей подзадачи.

Примеры

стандартный ввод	стандартный вывод
http://urlparse.ru	yes
https://urlparse	no
http://pravilny:adress//.ru	yes
http://.ru	yes

Замечание

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача D. НацВычМаш

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В рамках национальной программы «Цифровая экономика» знающими людьми было принято решение о построении Национальной Вычислительной Машины (сокращённо НВМ). Работать эта машина будет под управлением Национальной Операционной Системы и будет использоваться исключительно в национальных интересах.

Ивану Петровичу, сыну Петра Ивановича, программисту 4-го разряда, поручили реализовать часть базового функционала Машины, в рамках которого машина должна научиться вычислять, сколько раз число N, состоящее только из нулей и единиц, без ведущих нулей, можно делить на двойку до тех пор, пока результат остаётся чётным.

Формат входных данных

Вводится чётное положительное число N, состоящее из десятичных цифр 0 и 1, без ведущих нулей.

Формат выходных данных

Единственное целое число — ответ на задачу.

Система оценки

Подзадача 1 (баллы: 35)

$10 \leqslant N \leqslant 10^9$

Подзадача 2 (баллы: 15)

$10 \leqslant N \leqslant 10^{18}$

Подзадача 3 (баллы: 25)

$10 \leqslant N \leqslant 10^{1000}$

Подзадача 4 (баллы: 25)

$10 \leqslant N \leqslant 10^{10^6}$

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Примеры

стандартный ввод	стандартный вывод
10000	3
101010	0

Замечание

Обращаем ваше внимание, что число N даётся в десятичной системе счисления.

Пояснения к первому примеру:

10000 / 2 = 5000. Результат чётный, продолжаем делить.

5000 / 2 = 2500. Результат чётный, продолжаем делить.

2500 / 2 = 1250. Результат чётный, продолжаем делить.

1250 / 2 = 625. Результат нечётный, поэтому деление не включается в ответ.

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача E. Считалочки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Ура! Саша Вилкин празднует день рождения и по такому поводу принёс в школу конфеты. Раздав всем N одноклассникам по конфете, Саша обнаружил, что у него осталась лишь одна, которую он хочет отдать своему лучшему другу — Саше Ложкину (как же часто их дразнят из-за схожести имен...). Если Саша Вилкин просто отдаст последнюю конфету другу, то одноклассники начнут дразнить его ещё больше, а он очень этого не хочет! По этой причине Саша предложил одноклассникам определить «счастливчика» по считалочке.

В кругу стоят N школьников, пронумерованных с единицы по часовой стрелке. Саша знает M считалочек, притом i-я из них содержит в себе m_i тактов. Считать считалочку он начинает со школьника под номером 1 и идёт по часовой стрелке. Саша Вилкин хочет, чтобы последняя конфета досталась Саше Ложкину, который стоит под номером K.

Помогите Саше Вилкину выбрать нужную считалочку. Саша уверен, что знает достаточно считалочек, поэтому найдётся по меньшей мере одна, которая удовлетворит условию задачи.

Формат входных данных

В первой строке даны три целых положительных числа через пробел: N, M, K — количество школьников, количество считалочек, которые знает Саша Вилкин, и номер, под которым стоит Саша Ложкин, соотвественно. Во второй строке записаны M целых положительных чисел m_i — количество тактов в считалочке под номером i.

Формат выходных данных

Выведите единственное число — номер считалочки, которую стоит выбрать Саше Вилкину, чтобы последняя конфета досталась его другу. Нумерация считалочек начинается с 1. Если подходят несколько считалочек, выведите ту, у которой номер наименьший.

Система оценки

Подзадача 1 (баллы: 30)

$1 \leqslant N, M \leqslant 10000, 1 \leqslant K \leqslant N, 1 \leqslant m_i \leqslant 10^9$

Подзадача 2 (баллы: 50)

$1 \leqslant N, M \leqslant 10000, 1 \leqslant K \leqslant N, 1 \leqslant m_i \leqslant 10^9$

Подзадача 3 (баллы: 20)

$1 \leqslant N, M \leqslant 10000, 1 \leqslant K \leqslant N, 1 \leqslant m_i \leqslant 10^{18}$

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

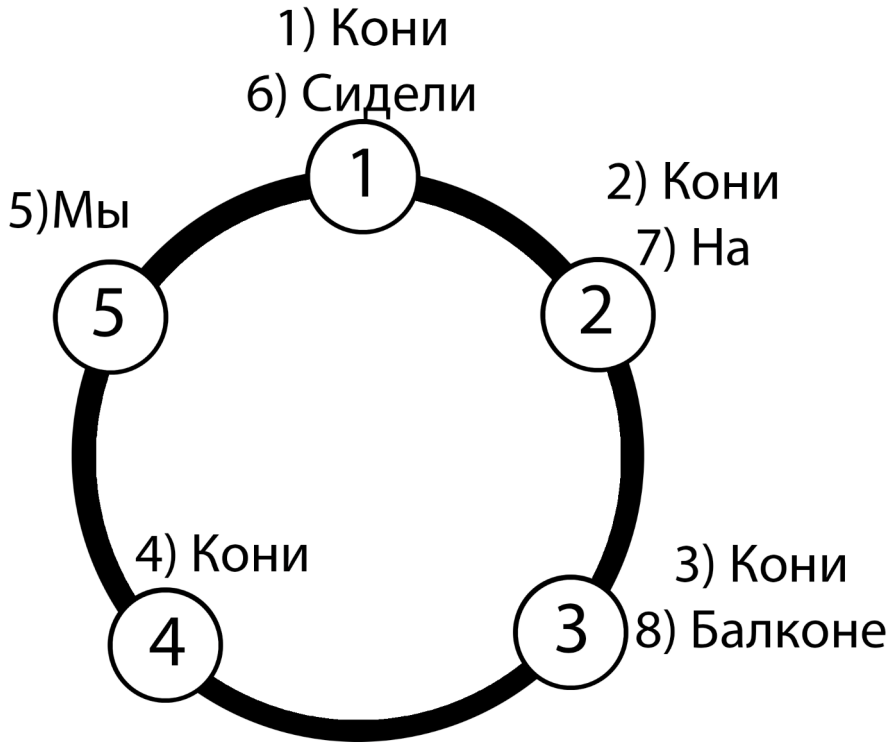
Пример

стандартный ввод	стандартный вывод
5 4 3 8 1 3 5	1

Замечание

В качестве считалочки для первого теста подходит простая детская «Кони, кони, кони, кони, мы сидели на балконе.»

Иллюстрация к примеру:



Первая считалочка на последнем такте указывает на ребёнка под номером 3, чего и добивался Саша Вилкин. Если бы Саша выбрал вторую считалочку, то попал бы на школьника под номером 1. Третья считалочка тоже подходит, но имеет больший номер. Четвертая считалочкауказала бы на ребенка с номером 5.

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача F. Яблоки в школу

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

У Евгения было a красных яблок, b жёлтых и c зелёных. Евгений хочет взять как можно больше яблок с собой в школу, но, чтобы он не взял слишком много, его мама поставила следующие условия:

1. Зелёных яблок должно быть меньше, чем жёлтых и красных вместе.
2. Красных яблок должно быть больше, чем удвоенное количество жёлтых минус количество зелёных.

Какое количество яблок каждого типа ему удастся взять так, чтобы суммарно их количество было максимально?

Формат входных данных

В первой строке вводятся три целых положительных числа: a, b и c — количество красных, жёлтых и зелёных яблок соответственно.

Формат выходных данных

Выведите три целых числа — количество красных, жёлтых и зелёных яблок, которое может взять Евгений, чтобы их сумма была максимальна. Если ответов несколько, выведите любой из них.

Система оценки

Подзадача 1 (баллы: 15)

$1 \leqslant a, b, c \leqslant 100$

Подзадача 2 (баллы: 20)

$1 \leqslant a, b, c \leqslant 1500$

Подзадача 3 (баллы: 30)

$1 \leqslant a, b, c \leqslant 10^5$

Подзадача 4 (баллы: 35)

$1 \leqslant a, b, c \leqslant 10^{10}$

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Примеры

стандартный ввод	стандартный вывод
2 5 4	2 2 3
10 9 8	10 8 8

Замечание

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача G. Заборный бизнес

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Кирилл хочет открыть свой бизнес в области...производства заборов! Как известно, в России сейчас это актуальная тема и только за 2019 год рынок заборов в нашей стране вырос на 146%! Кирилл проанализировал общие показатели рынка, исследовал спрос, целевую аудиторию, оценил конкуренцию и внешние факторы. Исходя из полученных данных, он решил, что наилучшим решением будет начать производить сетчатые заборы.

Мы предлагаем вам присоединиться к команде Кирилла (вас будет двое) и реализовать один из рисунков – Евросетчатый заборный рисунок.

Евросетчатый заборный рисунок — это циклический рисунок, который состоит из узелков и прутьев. От каждого узелка выходит ровно 4 диагональных прутика. $K + 1$ — длина каждого диагонального прутика. С двух сторон каждого прутика располагается узелок, из которого, в свою очередь, снова выходят диагональные прутики. **Для лучшего понимания евросетчатого заборного рисунка посмотрите на примеры ниже.**

Кирилл имеет заборную раму размера $N \times M$ и теперь хочет свить в ней из прутьев евросетчатый заборный рисунок. Для этого ему необходимо наглядно видеть сам рисунок. Помогите ему в этом.

Формат входных данных

В единственной строке вводятся три целых положительных числа через пробел: N, M — количество строк, столбцов, — и целое неотрицательное число K.

Формат выходных данных

Фрагмент евросетчатого заборного рисунка размера $N \times M$, в левом верхнем углу которого располагается узелок.

Формат вывода: ‘X’ — узелок; ‘/’, ‘\’ — прутики; ‘.’ — пустое пространство;

Система оценки

Подзадача 1 (баллы: 27)

$1 \leqslant N, M \leqslant 500; K = 0$

Подзадача 2 (баллы: 33)

$1 \leqslant N, M \leqslant 500; 0 \leqslant K \leqslant 1$

Подзадача 3 (баллы: 40)

$1 \leqslant N, M \leqslant 500; 0 \leqslant K \leqslant 500$

Для каждой подзадачи, кроме первой, необходимо, чтобы ваше решение корректно работало для предыдущих подзадач. Для первой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Примеры

стандартный ввод	стандартный вывод
5 9 1	X...X...X .\./.\./. ..X...X.. ./.\./.\. X...X...X
4 3 0	X.X .X. X.X .X.
3 14 2	X.....X.....X. .\.../.\.../.\ ..\./...\./...

Замечание

Длина прутьев, равная $K+1$, объясняется тем, что расстояние считается от центра символа 'X'.

Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к следующей задаче, возможно, она покажется тебе проще :)

Задача N. Гик-кафе

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Молодой предприниматель Богдан, владелец небольшого гик-кафе, путешествовал по миру в поисках новых идей. На этот раз наш герой решил попытать счастья в стране восходящего солнца — Японии.

И вот, прогуливаясь по ярким улицам Токио, Богдан понял, что проголодался. Посмотрев по сторонам, он заметил кафе с большой неоновой вывеской и решил зайти туда. Когда Богдан вошёл, он сильно удивился: гостей здесь обслуживали роботы-официанты! «Неплохая идея. Внедрю похожих роботов в своё кафе», — подумал Богдан, вот только удивляться было ещё рано. Он решил заказать «Максимально-хороший бургер». Как только наш предприниматель сделал заказ, из дверей кухни выехал очередной официант и немедленно поставил перед ошарашенным Богданом поднос с его заказом. «Впечатляюще быстро! Невероятно быстро!», — начал радовался про себя Богдан, но и в этот раз по-настоящему удивляться было рано! Богдан попробовал бургер и... буквально застонал от счастья, на глаза его навернулись слёзы: «Господи! Как это вкусно!». Не отдавая себе отчёт в своих действиях, Богдан ворвался на кухню заведения, чтобы узнать личность гениального повара и выведать его рецепт. Но то, что он увидел, повергло его в ещё больший шок.

За столом стоял робот с лопаткой, а перед ним в одну линию лежало множество упорядоченных ингредиентов (строка S). Как только прозвучал звуковой сигнал, робот с невероятной скоростью, за 2 секунды, собрал «Максимально-хороший бургер»! В замедленном темпе действия робота выглядели следующим образом: робот брал лопаткой ингредиент или слои уже лежащих друг на друге ингредиентов, затем **переворачивал** всё содержимое лопатки на **соседний** ингредиент. Следующим шагом робот повторял свой алгоритм с последними слоями бургера, которые образовались на **последнем шаге**. Иначе говоря, робот выбирал из множества ингредиентов отрезок и начинал собирать с какой-либо стороны бургер по принципу снежного кома — брал первый ингредиент, переворачивал и клал на второй, затем брал первый и второй, переворачивал и клал на третий, и так далее. В какой-то момент робот останавливался, брал получившиеся слои бургера и вкладывал их в булки. Заказ готов!

«В моё кафе непременно нужно точно такого же повара!» — подумал Богдан. Вспомнив, что его заказ назывался «Максимально-хороший бургер», предприниматель быстро нашёл листочек с точно таким же названием — вероятно, это рецепт. На листочке сказано, что хорошим бургером считается бургер, в котором два одинаковых ингредиента **не лежат друг на друге**; а «Максимально-хорошим» бургером называется хороший бургер, имеющий **максимально возможную длину**.

Вернувшись в родное кафе, Богдан сразу же принялся за разработку прототипа такого же робота-повара, только вот проблема оказалась в алгоритме определения, какой же бургер является максимально-хорошим. Поможете ему?

Формат входных данных

Вводится единственная непустая строка S, состоящая из строчных букв латинского алфавита — ингридиентов, которые робот видит перед собой.

Формат выходных данных

В первой строке выведите отрезок ингредиентов, из которых будет состоять «Максимально-хороший» бургер — выведите начало отрезка и его длину через пробел.

Во второй строке выведите направление, в котором роботу следует собирать бургер. Выведите «`Right`», если бургер необходимо собирать слева направо, и «`Left`» в противном случае.

Если «Максимально-хороших» бургеров несколько, выведите любой из них.

Система оценки

Подзадача 1 (баллы: 35)

$|S| \leqslant 100$

Для этой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Подзадача 2 (баллы: 25)

$|S| \leqslant 10^5$ **Строка не содержит двух одинаковых символов подряд.**

Для этой подзадачи необходимо, чтобы ваше решение корректно работало **на примерах из условия**.

Подзадача 3 (баллы: 40)

$|S| \leqslant 10^5$

Для этой подзадачи необходимо, чтобы ваше решение корректно работало на двух предыдущих.

Примеры

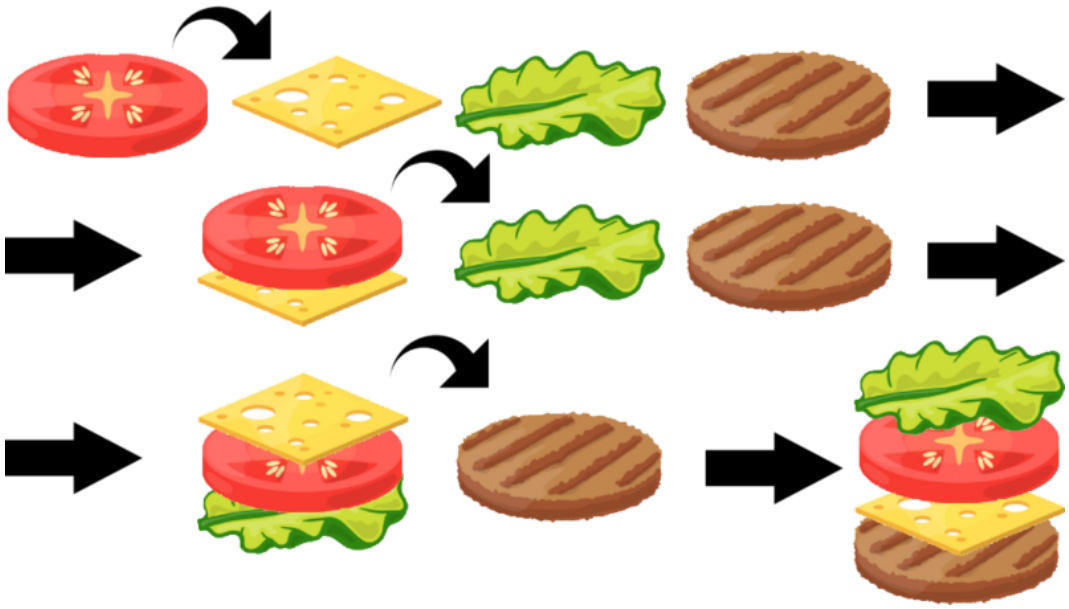
стандартный ввод	стандартный вывод
abbccdd	1 7 Right
aabcdeeef	2 6 Right
cccba	2 4 Left

Замечание

Пояснение к примерам из условия:

1. Ответ предполагает, что нужно взять отрезок `abbccdd` и готовить бургер слева направо. В итоге получается бургер `dcbabcd`, в котором ни один ингредиент не соседствует с себе подобным. Также этот отрезок является максимально длинным отрезком, на котором можно собрать хороший бургер.
2. Нужно взять отрезок `abcdee` и готовить бургер слева направо. В итоге получается бургер `ecabde`.
3. Нужно взять отрезок `ccba` и готовить бургер справо налево. В итоге получается бургер `cbac`.

На рисунке ниже наглядно показан процесс готовки бургера слева-направо.



Обратите внимание: чтобы набрать не 0 баллов по этой задаче, не обязательно решать её в полных ограничениях. Сдай несколько подзадач и переходи к предыдущей задаче, возможно, ты осознаешь что она проще :)

Вузовско-академическая олимпиада по информатике

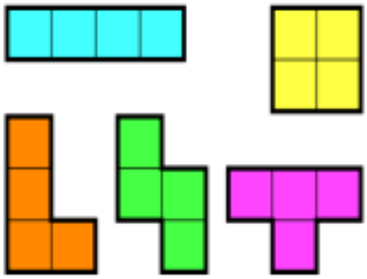
2019-2020 учебный год

Задания второго (заключительного) этапа

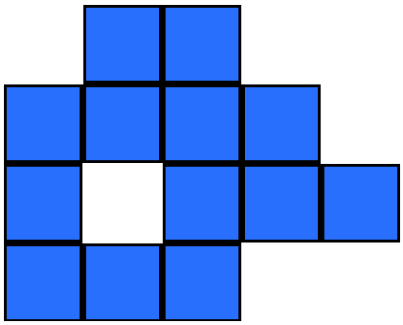
Задача A. Плохой тетрис

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

А вы знали, что слово «Тетрис» образовано от слов «тетрамино» и «теннис»? Если с теннисом все более-менее знакомы, то вот что такое тетрамино знают немногие. Тетрамино — плоские геометрические фигуры, состоящие из четырёх квадратов, соединённых сторонами.



Тетрамино являются подмножеством полимино. Разница только в том, что если тетрамино состоят всегда из 4 квадратов, то полимино — из одного и более одноклеточных квадртатов, соединенных по их сторонам.



На рисунке изображён пример полимино; обратите внимание, что полимино может содержать дырки.

Цель игры «Тетрис» — удалить как можно блоков так, чтобы они не вышли за пределы поля. В этой задаче предлагается сделать противоположное — построить башню как можно выше. Землей для строительства башни является бесконечная прямая, ограничений по ширине башни нет. В игре действует простая физика — каждая фигура падает вниз под действием гравитации, и она будет стоять, если хотя бы один из её блоков стоит на земле или другой не падающей фигуре, даже если центр масс не находится над опорой.

Посчитайте, какой максимальной высоты башню можно построить из данных фигур полимино, если фигуры можно поворачивать на 90 и 180 градусов в обе стороны, смещать по горизонтали и использовать в любом порядке.

Формат входных данных

В первой строке указано целое число N — количество полей ($1 \leqslant N \leqslant 100$).

Далее идет N блоков строк с описанием полей, на которых представлены фигуры. В первой строке каждого блока указывается число K_i — количество строк, которое занимает поле ($1 \leqslant K_i \leqslant 100$). Затем идут K_i строк — поле, на котором изображено полимино. Все K_i строк имеют одинаковую длину от 1 до 100 и состоят только из знаков «_» (код 95) и «#» (код 35) (без кавычек), где знак «_» означает пустую клетку, а знак «#» — квадрат (элемент полимино). Гарантируется, что на каждом поле есть ровно одна фигура полимино.

Обратите внимание на необычные ограничения в группах тестов, указанные в поле «Система оценки».

Формат выходных данных

Выведите максимальную высоту башни, которая может получится из данных фигур полимино.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	32	см. ниже	
2	30	см. ниже	1
3	38	см. ниже	1, 2

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Подзадача 1 (баллы: 32)

Гарантируется, что полимино могут быть только квадратами, т.е. фигурами размера $W \times W$.

Подзадача 2 (баллы: 30)

Гарантируется, что полимино могут быть только прямоугольниками, т.е. фигурами размера $W \times H$.

Подзадача 3 (баллы: 38)

Полимино могут быть произвольной формы.

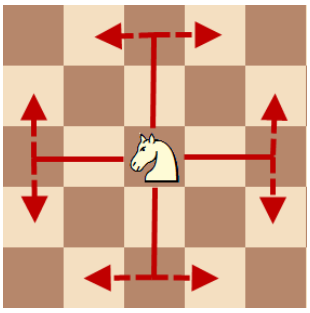
Примеры

стандартный ввод	стандартный вывод
2 3 _____ ___## ___## 2 _##_ _##_	4
2 5 _____ ####_ ####_ ####_ ####_ 2 ## ##	6

Задача B. Он вам не конь 3

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Игорь — игрок в шахматы. После очередной тренировки Игорь со своим напарником решили сыграть, имея только одного коня, но с использованием не одной шахматной доски, а сразу нескольких. Напомним, что конь ходит буквой «Г» во всех направлениях так, что сначала двигается на 2 клетки по одной оси и на 1 — по другой. Все варианты одного хода для шахматного коня изображены на рисунке:



На прямоугольном столе, размеченном квадратной сеткой, Игорь разложил N досок, каждая из которых — прямоугольник размером не менее, чем 3×3 (не менее 3 клеток по каждой из осей). Доски расположены так, что их стороны параллельны сторонам стола, а углы доски — строго в узлах квадратной сетки. Кроме того, никакие две доски не касаются друг друга ни сторонами, ни даже углами. В центре стола выбрано начало координат, сами координаты по модулю не превосходят M.

В некоторую клетку одной из этих N досок Игорь поставил коня. Необходимо посчитать, сколько досок может посетить конь, последовательно совершая свои ходы. При этом ходить можно только по одной доске, а также между ними, если это можно сделать за один ход. Разрешается посещать ранее пройденные клетки и доски.

Формат входных данных

В первой строке через пробел даны два целых числа N и M ($1 \leqslant N \leqslant 5 \cdot 10^4$, $2 \leqslant M \leqslant 10^9$).

Во второй строке через пробел даны два целых числа x и y — координаты левого нижнего угла клетки, в которой находится конь ($-M \leqslant x, y < M$).

В каждой из следующих N строк через пробел даны по 4 целых числа: x_{Li}, y_{Li}, x_{Ri}, y_{Ri}, задающие очередную шахматную доску, где $(x_{Li}; y_{Li})$ — это координаты левого нижнего угла доски, а $(x_{Ri}; y_{Ri})$ — правого верхнего ($-M \leqslant x_{Li} < x_{Ri} \leqslant M$, $-M \leqslant y_{Li} < y_{Ri} \leqslant M$). Гарантируется, что размер доски по каждому из измерений не меньше 3, то есть $x_{Ri} - x_{Li} \geqslant 3$ и $y_{Ri} - y_{Li} \geqslant 3$, и что никакие две доски не касаются друг друга ни сторонами, ни даже углами.

Гарантируется, что клетка с конём будет находиться на некоторой доске, то есть для некоторого i выполняется $x_{Li} \leqslant x < x_{Ri}$ и $y_{Li} \leqslant y < y_{Ri}$.

Формат выходных данных

В единственной строке выведите одно целое число — количество досок, которые может посетить конь, включая доску, с которой он начинает своё путешествие.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	16	$N \leqslant 2, M \leqslant 100$	
2	18	$N \leqslant 100, M \leqslant 100$	1
3	20	$N \leqslant 10^3, M \leqslant 10^9$	1, 2
4	25	$N \leqslant 5 \cdot 10^4, M \leqslant 10^5$	1, 2
5	21	$N \leqslant 5 \cdot 10^4, M \leqslant 10^9$	1, 2, 3, 4

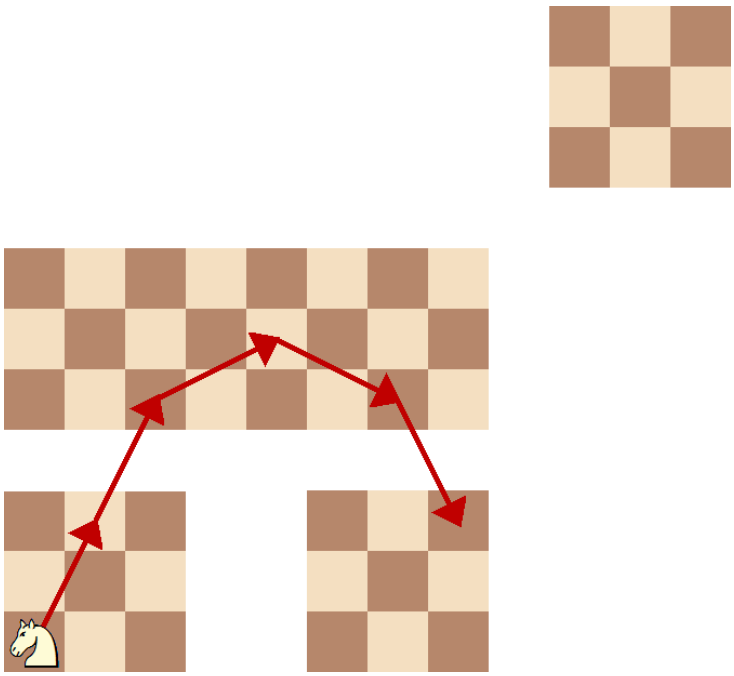
Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Примеры

стандартный ввод	стандартный вывод
4 12 0 0 0 0 3 3 5 0 8 3 0 4 8 7 9 8 12 11	3
2 10 -1 -1 -2 -2 1 1 -2 2 1 5	1

Замечание

Иллюстрация к первому примеру:



Задача C. Правильные печеньки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В некотором крае производством печенья занимается фирма Хомяшкино. Изделия этой компании знамениты оригинальной формой: каждая печенька представляет из себя правильный многоугольник, а фирменные упаковки для печенья имеют форму цилиндра. Печенья складывают в упаковку стопкой и делают такого размера, чтобы уголки помещались вплотную к боковой поверхности цилиндра.

Конвейер, складывающий печенья по порядку в цилиндрическую упаковку, не может поворачивать печеньки и помещает их все под одним и тем же углом. Таким образом, у каждой печеньки один из уголков оказывается на прямой, общей для всех печенек (эта прямая обозначена на рисунке буквой p).

В одной упаковке покупатель может обнаружить печенья с разным количеством уголков. Для возникновения эффекта неожиданности печеньки стараются складывать в таком порядке, чтобы при взгляде со стороны открывающейся части упаковки (т.е. верхнего основания цилиндра) было видно наименьшее количество печенек.

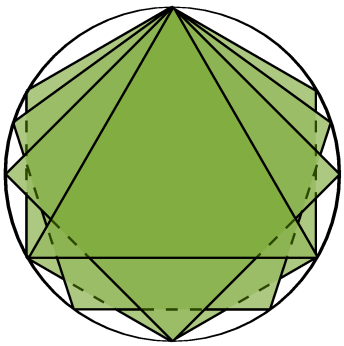


Рис. 1: Вид снизу

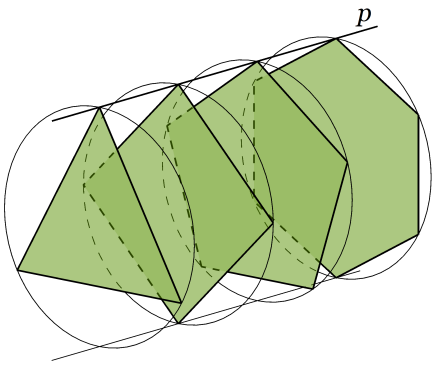


Рис. 2: Вид сбоку

Главный технолог собирается запрограммировать конвейер на упаковывание набора из N печений с количествами сторон a_1, a_2, …, a_N у первой, второй, …, N-й печеньки соответственно. Помогите ему выяснить, какое наименьшее количество печенек может быть видно при взгляде сверху на открытую упаковку с данным набором печений?

Формат входных данных

В первой строке указано целое число N ($1 \leqslant N \leqslant 50000$).

В каждой из следующих N строк указано по одному целому числу a_i ($3 \leqslant a_i \leqslant 10^5$).

Формат выходных данных

Требуется вывести единственное целое число — наименьшее количество видимых печенек при их оптимальном расположении.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	18	$N \leqslant 2$	
2	40	$N \leqslant 1000$	1
3	42	$N \leqslant 50000$	1, 2

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Пример

стандартный ввод	стандартный вывод
4 3 4 5 6	3

Задача D. Киберспортивный турнир

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2.5 секунд
Ограничение по памяти:	256 мегабайт

Илья является администратором самой популярной многопользовательской киберспортивной дисциплины. Руководство каждый раз отправляет его в командировки, чтобы администрировать оффлайн турниры. Илья — профессионал своего дела, поэтому руководство уверено в том, что все соревнования, которые будет администрировать Илья, пройдут без технических неполадок!

На одном из крупнейших турниров Илья вновь был администратором и имел ник «`admin`» (без кавычек). Конечно же наш герой очень ответственный и пришел на работу заранее и сразу же вошёл в игру, а также собирается сидеть на ней допоздна, не отвлекаясь.

Как известно, организация турниров — вещь очень сложная и долгая. Пока все участники соревнования подключатся, некоторые из них могут отойти ненадолго по разным причинам и не быть в сети, и задержка происходит зачастую как раз из-за них. Ну а пока происходит длительное подключение всех игроков, в чате все желают друг другу удачи и веселья в предстоящем матче. От смертельной скуки Илье стало интересно, какую удачу имеет каждый игрок в матче. Илья считает, что *удача* — положительное вещественное число, имеющееся у каждого игрока.

Сообщением *формата GLHF* будем называть сообщение, состоящее только из букв алфавита {«`gl`», «`hf`»} (без кавычек). Иначе говоря, сообщение формата GLHF — это строка, содержащая только пары букв «`gl`» и «`hf`». Например: «`glglglhfhf`», «`hf`», «`hfgl`». В частности, все сообщения формата GLHF имеют чётную длину. Игроки посылают сообщения только формата GLHF.

Каждый игрок (включая администратора) в самом начале имеет удачу, равную единице. Илья считает что удача игрока увеличивается в A раз каждый раз, когда он в чате видит пожелание удачи («`gl`» — good luck), и сокращается в B раз каждый раз, когда он видит пожелание провести хорошо время и повеселиться («`hf`» — have fun) (как известно, в напряжённом матче не стоит расслабляться и веселиться). Если в сообщении «`gl`» и/или «`hf`» написано несколько раз, то удача увеличивается в A раз и уменьшается в B раз за каждую пару символов «`gl`» и «`hf`» соответственно.

Обдумав всё это, Илья захотел сравнить удачи игроков, и хоть времени на это у него предостаточно, он поручил вам реализовать небольшую утилиту.

Ваша маленькая утилита получает на вход корректные логи — сообщения формата: «`[hh:mm:ss] AUTHOR_NICK: /command`» (без кавычек), где `[hh:mm:ss]` — время в формате `[часы:минуты:секунды]`, `AUTHOR_NICK` — ник игрока, `command` — команда. Возможных команд несколько:

- `/j`, `/join` — игрок `AUTHOR_NICK` подключается к игре. После подключения игрок может видеть все последующие личные и общие сообщения в чате. На данную команду сервер может ответить:
  - «`Successful`» — если на момент выполнения команды игрок не находился в игре.
  - «`Failed: AUTHOR_NICK is already on the server`» — если игрок уже игре.
- `/q`, `/quit` — игрок `AUTHOR_NICK` отключается от игры. После отключения игроку не приходят никакие последующие личные или общие сообщения, пока он не войдёт повторно. Выход с сервера не меняет удачу игрока, и после повторного захода удача будет такой, какой она была при выходе. Сервер должен ответить:
  - «`Successful`»
- `/o`, `/online` — игрок `AUTHOR_NICK` хочет узнать, сколько сейчас человек онлайн. Сервер должен вывести:
  - «`There are X players on the server`» — где вместо X написано количество людей на сервере (включая администратора).

- `/s [glhf]`, `/say [glhf]` — игрок `AUTHOR_NICK` пишет в общий чат сообщение формата GLHF, по длине не превосходящее M. Это сообщение увидят все подключенные игроки (включая администратора). Такие сообщения не влияют на удачу самого автора. Сервер должен ответить:
  - «`AUTHOR_NICK: [glhf]`» — вывод имени игрока и его сообщения.
- `/w<RECIPIENT_PLAYER> [glhf]`, `/write<RECIPIENT_PLAYER> [glhf]` — игрок `AUTHOR_NICK` пишет личное сообщение игроку `RECIPIENT_PLAYER` формата GLHF, по длине не превосходящее M. В этом случае сообщение видит только `RECIPIENT_PLAYER`, но только если он подключен к игре. При отсутствии адресата на сервере никаких сообщений об ошибке не возникает. Такие сообщения не влияют на удачу самого автора. Игрок не может написать сообщение самому себе. Сервер должен ответить:
  - «`AUTHOR_NICK (to RECIPIENT_PLAYER): [glhf]`» — вывод имени игрока, имени получателя и сообщения.
- `/c<PLAYER>`, `/compare<PLAYER>` — данная команда доступна только для игрока `admin`. Команда предназначена для сравнения удачи администратора с игроком `PLAYER`. Сервер может ответить на данную команду следующее:
  - «`Failed: you have no such rights`» — если `AUTHOR_NICK` не является `admin` (Наивысший приоритет проверки).
  - «`Failed: PLAYER is not on the server`» — если `PLAYER` в данный момент не находится в игре.
  - «`PLAYER is a loser`» — если удача `admin` больше, чем удача игрока `PLAYER`
  - «`PLAYER is good`» — если удача `admin` равняется удаче игрока `PLAYER`
  - «`PLAYER is a lucky guy`» — если удача `admin` меньше, чем удача игрока `PLAYER`
- `/c<PLAYER1>-<PLAYER2>`, `/compare<PLAYER1>-<PLAYER2>` — данная команда так же доступна только для игрока `admin`. Команда предназначена для сравнения удачи игрока `PLAYER1` с удачей игрока `PLAYER2`. Сервер может ответить на данную команду следующее:
  - «`Failed: you have no such rights`» — если `AUTHOR_NICK` не является `admin` (Наивысший приоритет проверки).
  - «`Failed: PLAYER1 is not on the server`» — если `PLAYER1` в данный момент не находится в игре. (Следующий по приоритету проверки)
  - «`Failed: PLAYER2 is not on the server`» — если `PLAYER2` в данный момент не находится в игре.
  - «`PLAYER2 is a loser`» — если удача `PLAYER1` больше, чем удача игрока `PLAYER2`
  - «`Their luck is equal`» — если удача `PLAYER1` равняется удаче игрока `PLAYER2`
  - «`PLAYER2 is a lucky guy`» — если удача `PLAYER1` меньше, чем удача игрока `PLAYER2`

За каждую встреченную подстроку «`gl`» и «`hf`» удача игрока увеличивается в A, либо уменьшается в B раз соответственно. Стоит заметить что удача игрока не сбрасывается после отключения и повторного подключения к игре.

Ваша задача состоит в реализации данной утилиты!

Формат входных данных

В первой строке через пробел вводятся целые числа N, M, A, B — количество команд, которые получит сервер, максимальное количество символов в сообщении, множитель удачи игрока, когда тот видит сообщение «`gl`», и делитель удачи игрока, когда тот видит сообщение «`hf`», соответственно ($1 \leqslant N \leqslant 86400$, $2 \leqslant M \leqslant 10$, $1 \leqslant A, B \leqslant 10^9$).

Далее идут N строк, в которых, расположены логи сервера в хронологическом порядке, каждая строка имеет следующий формат:

«`[hh:mm:ss] AUTHOR_NICK: /command`» (без кавычек). Временные метки состоят из часов `hh` (от 0 до 23), минут `mm` (от 0 до 59) и секунд `ss` (от 0 до 59); каждое из этих чисел имеет ровно 2 цифры и при необходимости дополняется ведущим нулём. Временные метки не повторяются и идут строго по возрастанию. Ник игрока `AUTHOR_NICK`, а также все ники в аргументах команд `/write` и `/compare` — строки, состоящие из больших и маленьких латинских букв, цифр, а также символа «_» (код 95); длина каждого ника не превосходит 8 символов.

Логи турнира корректные, гарантируется отсутствие входящих команд от участников, не подключённых к игре, кроме команд `/join`. Все сообщения придерживаются формата GLHF и имеют длину не более M символов.

Изначально считается, что на сервере нет никого, кроме администратора, имеющего ник «`admin`». Администратор никогда не покидает сервер.

Гарантируется, что если операция сравнения удачи двух игроков показала, что их удача различается, то их значения удачи различаются хотя бы в 1.0001 раза.

Формат выходных данных

Вывод должен содержать N строк — ответы сервера на запросы, где i строка соответствует ответу сервера на запрос в $i + 1$ строке входных данных.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	24	$N \leqslant 8; M = 2; A = B; A, B \leqslant 5$; нет доступа к команде /q	
2	18	$N \leqslant 8; M \leqslant 4; A, B \leqslant 5$	1
3	18	$N \leqslant 100; M \leqslant 10; A = B; A, B \leqslant 1000$	1
4	22	$N \leqslant 500; M \leqslant 10; A, B \leqslant 10^9$	1, 2, 3
5	18	$N \leqslant 86400; M \leqslant 10; A, B \leqslant 10^9$	1, 2, 3, 4

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Обратите внимание на необычное ограничение в первой подзадаче, где гарантируется отсутствие команд `/quit` во входных данных.

Пример

стандартный ввод	стандартный вывод
18 4 2 4 [00:00:23] admin: /online [00:01:02] j0e: /j [00:01:15] j0e: /s hfhf [00:01:17] j0e: /q [00:01:18] admin: /write<j0e> hfhf [00:03:25] Tea_: /join [00:04:11] j0e: /join [00:07:54] Tea_: /j [00:08:30] admin: /say glgl [00:08:37] admin: /write<j0e> hfhf [00:09:00] Tea_: /compare<j0e> [00:13:41] admin: /c<j0e>-<Tea_> [00:13:44] j0e: /quit [00:13:57] admin: /compare<j0e> [00:27:49] j0e: /j [00:28:01] j0e: /s hfhf [00:30:19] admin: /compare<j0e> [00:30:43] admin: /compare<Tea_>-<j0e>	There are 1 players on the server Successful j0e: hfhf Successful admin (to j0e): hfhf Successful Successful Failed: Tea_ is already on the server admin: glgl admin (to j0e): hfhf Failed: you have no such rights Tea_ is a lucky guy Successful Failed: j0e is not on the server Successful j0e: hfhf j0e is a lucky guy Their luck is equal

Замечание

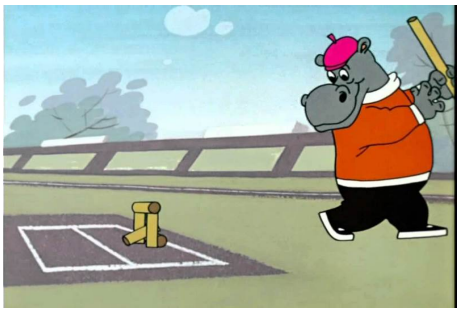
Пояснение к примеру:

- После команды «[00:01:15] j0e: /s hfhf» удача admin сократилась в 4^2 раза. Удача j0e не меняется.
- После команды «[00:01:18] admin: /write<j0e> hfhf» удача игрока j0e не изменилась, так как он отключен от игры. Удача admin также не изменилась.
- После команды «[00:08:30] admin: /say glgl» удача j0e и Tea_ увеличивается в 2^2 раза. Удача admin не меняется. Теперь удачи у игроков следующие: admin: $1/4^2$; j0e: 2^2; Tea_: 2^2.
- После команды «[00:08:37] admin: /write<j0e> hfhf» удача j0e сокращается в 4^2 раза и становится равной $2^2/4^2$. Удача admin не меняется.
- Выполнение команды «[00:09:00] Tea_: /compare<j0e>» вызовет ошибку, потому что Tea_ не имеет достаточно прав для её выполнения.
- Команда «[00:13:41] admin: /c<j0e>-<Tea_>» сравнивает удачи игроков j0e и Tea_, их удача равна $2^2/4^2$ и 2^2 соответственно. У Tea_ удача выше, поэтому ответ сервера: «Tea_ is a lucky guy».
- Выполнение команды «[00:13:57] admin: /compare<j0e>» вызовет ошибку, так как игрок j0e не подключен к игре. Ответ сервера: «<Failed: j0e is not on the server».
- После команды «[00:28:01] j0e: /s hfhf» удача игроков admin и Tea_ сокращается в 4^2 раза. Удача j0e не меняется.
- Командой «[00:30:19] admin: /compare<j0e>» admin желает сравнить свою удачу с удачей игрока j0e. Их удачи составляют $1/4^4$ и $2^2/4^2$ соответственно. Удача j0e больше, поэтому ответ сервера: «<j0e is a lucky guy»
- Командой «[00:30:43] admin: /compare<Tea_>-<j0e>» admin желает сравнить удачу игрока Tea_ с удачей игрока j0e. Их удачи равны $2^2/4^2$ и $2^2/4^2$ соответственно. Их удача равна, поэтому ответ сервера: «Their luck is equal»

Задача E. Городки и роботы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Городки — русская спортивная игра. В ней нужно с помощью меткого броска биты, шеста или другой палки уничтожать так называемые «городки» — постройки из маленьких деревянных брусков.



Но время не стоит на месте, и теперь уничтожением построек занимаются не люди, а роботы. Правила игры очень просты. На огромном поле расположены $2N$ городков с целочисленными координатами, объединённых в пары (то есть, всего N пар городков), среди которых нет двух городков с одинаковыми координатами. В наличии имеется N роботов — машинок, работающих на масле. В начале игры робот высаживается на любой неразрушенный городок и тут же разбивает его, сам робот при этом остаётся целым. Затем этот робот должен доехать до соответствующего второго городка из пары. Маршрут может быть любым, в том числе прямой, ломаной или кривой линией. Доехав до второго городка из пары, робот тихонько самоуничтожается, разрушая этот и только этот городок. После уничтожения робота высаживается новый робот, разрушая один из городков при высадке, затем он доезжает до второго городка из пары и самоуничтожается, разрушая этот городок. Это происходит до тех пор, пока все N роботов не самоуничтожены, и все $2N$ городков не разрушены. Путь робота должен не проходить ни через один городок кроме тех двух, которые он должен разрушить.

Каждый робот имеет конструктивный недостаток: во время перемещения из него постоянно выливается масло. При движении робота за ним остаётся масляный след в виде непрерывной кривой линии, которая в точности повторяет его маршрут. При попадании на масляный след робот тут же выходит из строя, делая невозможным успешное завершение игры, поэтому пути роботов должны не пересекаться.

Чтобы сделать игру более захватывающей, на поле начертили масляный след в виде прямоугольника с координатами вершин $(0;0)$, $(W;0)$, $(W;H)$, $(0;H)$. Городки могут располагаться как вне прямоугольника, так и внутри, и на самом прямоугольнике. Робот никогда не ломается при высадке на городок, даже если тот был прямо на границе прямоугольника, но робот сломается, если наедет на начерченный след. Городок всегда разбивается, когда до него добирается робот, даже если этот городок стоял на границе прямоугольника.

Считается, что размеры роботов и городков и толщина масляных следов настолько малы, что роботов и городки можно считать точками, а следы — кривыми линиями.

Игра считается проваленной, если один из роботов ломается, не разбив второй городок из пары, до которого должен был доехать. Если все N пар городков разрушены N роботами, то игра считается выигранной.

Требуется определить, можно ли так составить маршруты роботов, чтобы разбить все городки.

Формат входных данных

В первой строке через пробел записаны три целых числа: N, W и H — количество пар городков, ширина прямоугольника и высота прямоугольника соответственно ($1 \leqslant N \leqslant 10^5$, $1 \leqslant W, H \leqslant 10^8$).

В каждой их следующих N строк через пробел записаны по 4 целых числа x_1, y_1, x_2, y_2, где $(x_1; y_1)$ — координаты первого городка из пары, а $(x_2; y_2)$ — координаты второго городка из этой же пары ($-10^8 \leqslant x_1, y_1, x_2, y_2 \leqslant 10^8$).

Гарантируется, что координаты всех городков попарно различны; это относится как к городкам из одной пары, так и к городкам из разных пар.

Формат выходных данных

В качестве ответа в единственной строке требуется вывести «**YES**» (без кавычек), если можно разбить все городки с помощью N роботов, и «**NO**» (без кавычек) в противном случае.

Система оценки

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	14	$N = 1$	
2	10	$N \leqslant 2$	1
3	20	$N \leqslant 3$	1, 2
4	34	$N \leqslant 500$	1, 2, 3
5	22	$N \leqslant 10^5$	1, 2, 3, 4

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Примеры

стандартный ввод	стандартный вывод
1 2 2 -1 1 3 1	YES
3 4 4 0 0 3 3 4 0 1 3 2 4 2 0	YES
3 4 4 0 0 4 4 4 0 0 4 2 4 2 0	NO

Задача F. Веб Котики

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Илья работает web-разработчиком, и, как положено web-разработчику, перед началом работы он общается с web-дизайнером, который показывает ему желаемый макет web-страницы. В этот раз дизайнер попросил Илью сделать web-страницу ширины W пикселей и разместить на ней в один ряд N картинок с котиками ширины A пикселей каждая, а расстояния между изображениями и краями экрана должны быть одинаковыми и положительными. Иначе картинки будут занимать всю ширину экрана и выглядеть неэстетично. Кроме того, картинки вместе с отступами должны занимать всю ширину до единого пикселя. И так как перед котиками не устоит никто, Илья сразу принялся за работу.

Спустя некоторое время перед ним возникла проблема: чтобы эту web-страницу все пользователи видели корректно, все заданные параметры должны быть целочисленными! Иначе пользователи разных браузеров будут видеть эту страницу по-разному. Ширину страницы мы изменить не можем, а вот с шириной картинок и расстояниями между этими картинками нужно что-то делать. Илья старается найти такие значения, которые были бы целочисленными, и чтобы ширина картинок была максимально близка к требованию дизайнера, но что-то у него не получается. Может, получится у вас?

Формат входных данных

В единственной строке через пробел дано три целых числа W, N, A — ширина веб страницы, количество картинок в одном ряду и желаемая дизайнером ширина картинок соответственно ($2 \leqslant W \leqslant 10^9$, $1 \leqslant N < W$, $1 \leqslant A < W$).

Формат выходных данных

Выведите единственное целое число — новую ширину картинок, наиболее близкую к желаемой ширине дизайнера A, либо «**NO**» (без кавычек), если такого значения нет. Если подходящих наиболее близких значений несколько, выведите наибольшее из них.

Система оценки

В данной задаче 20 тестов, каждый тест оценивается в 5 баллов. Тестовые примеры не входят в это число и оцениваются в 0 баллов. Вам будет сообщён только суммарный балл по всем тестам. Баллы начисляются только в случае, если решение **корректно работает на примерах из условия.**

Примеры

стандартный ввод	стандартный вывод
1200 10 70	65
120 12 5	NO
101 50 100	1

Задача G. Любимые бутерброды

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Валя очень любит бутерброды с колбасой. Особенно когда они нарезаны как на картинке.

По рецепту Вали для одного бутерброда нужно A грамм хлеба и B грамм колбасы. В ближайшем магазине продаются буханки хлеба по X грамм и палки колбасы по Y грамм. Когда Вале не хватает ингредиентов на очередной бутерброд, он идет в магазин и покупает наименьшее число буханок и палок, но столько, чтобы ему хватило на один бутерброд. Изначально у него ничего нет. Известно, что Валя съел ровно N бутербродов. Сколько раз он ходил в магазин?

Формат входных данных

В единственной строке даются числа A, B, X, Y, N в соответствующем порядке, разделённые пробелом. ($1 \leqslant A, B \leqslant 100$; $1 \leqslant X, Y, N \leqslant 10^9$)

Формат выходных данных

В единственной строке нужно указать количество Валиных походов в магазин.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	22	$A, B, X, Y, N \leqslant 100$	
2	39	$A, B, X, Y \leqslant 100, N \leqslant 10^9$	1
3	39	$A, B \leqslant 100, X, Y, N \leqslant 10^9$	1, 2

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Примеры

стандартный ввод	стандартный вывод
30 20 90 30 4	3
5 6 7 72 100	74
99 100 37 47 1000000000	1000000000

Замечание

Пояснение к первому примеру:

Изначально у Вали ничего нет, поэтому он идет в магазин за одной буханкой хлеба и одной палкой колбасы, после чего у него становится 90г хлеба и 30г колбасы на руках. Из этого можно сделать один бутерброд, после чего остаётся 60г хлеба и 10г колбасы. На следующий бутерброд не хватает колбасы, поэтому Валентин идет второй раз в магазин за одной палкой колбасы, после чего у него на руках 60г хлеба и 40г колбасы. Из этого можно сделать 2 бутерброда, после чего никаких продуктов не остаётся. Для того чтобы сделать 4-й бутерброд, Валя в третий раз идёт в магазин за хлебом и за колбасой и готовит его.

Вузовско-академическая олимпиада по информатике

2019-2020 учебный год

Разбор заданий второго (заключительного) этапа

Задача A: Плохой тетрис

Нужно составить самую высокую башню из множества полимино. При этом считается, что башня не падает под действием гравитации.

Подзадача 1: Квадраты

Можно просто ставить квадраты друг на друга любым способом. Ответом будет сумма длин сторон всех квадратов. Необходимо научиться считать длину стороны квадрата. Для этого можно найти строчку, в которой есть хотя бы одна клетка квадрата, и найти в этой строчке индекс первой такой клетки и последней такой клетки. Разность индексов, увеличенная на 1, будет равна ширине.

Подзадача 2: Прямоугольники

Стоит посчитать ширину и высоту прямоугольника. Ширина считается способом из первой подзадачи, высота равна количеству непустых строк. Далее, поскольку требуется построить максимально высокую башню, — нужно взять максимум из высоты и ширины и прибавить к ответу.

Подзадача 3: Полимино

В случае с произвольным полимино посчитать высоту и ширину приведённым выше способом не удастся. Однако, есть альтернативный способ. Поскольку известно, что на каждом поле есть ровно одна фигура полимино, достаточно найти самую верхнюю, самую нижнюю, самую правую и самую левую клетку, принадлежащую полимино. Исходя из их координат можно посчитать ширину и высоту полимино. Далее, аналогично подзадаче 2, необходимо взять максимум из высоты и ширины.

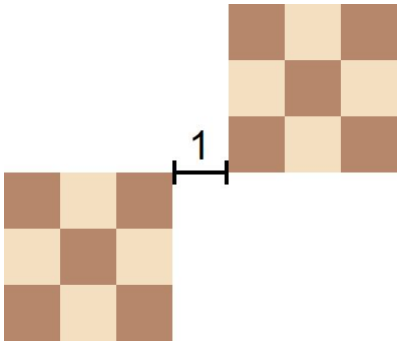
Задача B: Он вам не конь 3

На клетчатой плоскости разместили N шахматных досок разного размера, не касающихся ни сторонами, ни углами. В одну из клеток поставили шахматного коня. Определить, сколько досок он может посетить

Подзадача 1: N ≤ 2, M ≤ 100

Для начала разберем крайний случай, общий для всех подзадач: если конь начал в центральной клетке доски 3 на 3, то он никуда не может сходить. В этом случае ответ 1.

Иначе, конь может посетить все клетки стартовой доски и все клетки любой доски, куда он может попасть (кроме центральной клетки доски 3 на 3). Следовательно, нужно проверить, существует ли ход из клетки первой доски в клетку второй доски. Это можно сделать разными способами, например, перебрав все клетки первой доски и попробовав сходить из каждой. Или можно посчитать расстояние между границами досок, такой ход существует, если расстояние равно 1, то есть, расстояние между ближайшими двумя точками из разных досок равно 1.



Подзадача 2: N ≤ 100, M ≤ 100

Чтобы решить эту подзадачу, можно построить граф на клетках всего поля, лежащих на одной из досок (их не более 40000), соединить ребрами пары клеток, между которыми есть ход конем, и запустить поиск в глубину из стартовой клетки. После чего посчитать количество досок, которые достиг поиск в глубину.

Подзадача 3: N ≤ 1000, M ≤ 10^9

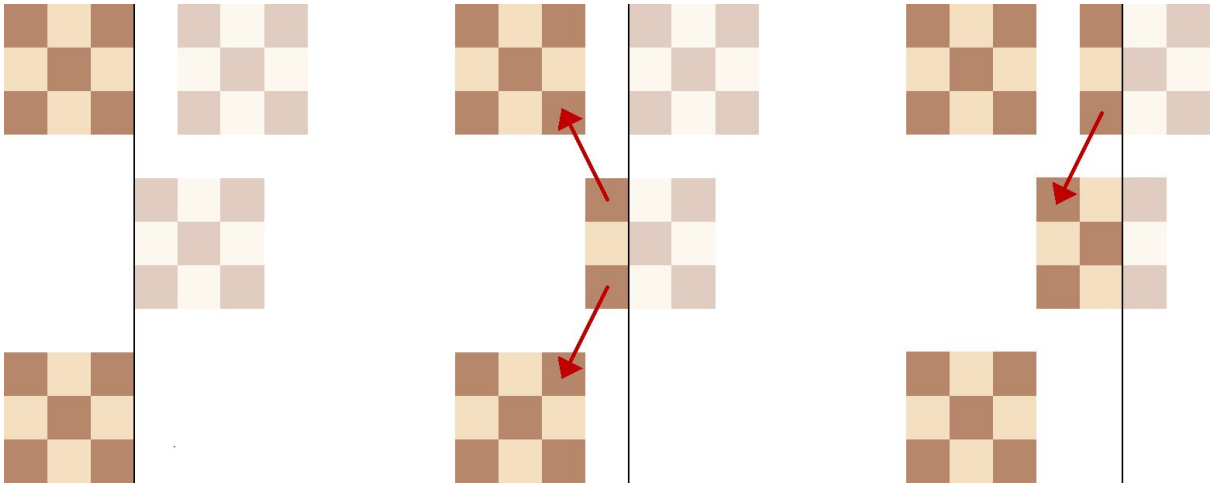
Для этой подзадачи необходимо для пары досок уметь считать, есть ли ход из одной в другую. Это можно сделать, вычислив расстояние между прямоугольниками.
Чтобы решить подзадачу, построим граф на досках, соединив ребрами пары досок, между которыми есть ход. После чего запустим поиск в глубину из начальной доски и посчитаем количество посещенных вершин в графе.

Асимптотика этого решения — $O(N^2)$, так как для построения ребер в графе нужно перебрать все пары вершин.

Подзадача 4: N ≤ 50000, M ≤ 100000

Покажем, что граф, построенный в предыдущей подзадаче, имеет не более $4N$ ребер. Разделим ребра на 2 группы: вертикальные и горизонтальные; вертикальные ребра будут соответствовать ходу конем на 2 клетки по вертикали и 1 по горизонтали, горизонтальные ребра — ходу на 2 клетки по горизонтали и 1 по вертикали. Покажем, что вертикальных ребер не более $2N$ для горизонтальных все аналогично.

Начнем с поля, на котором нет ни одной клетки доски, и будем добавлять столбцы слева направо. Чтобы добавить столбец, добавим все клетки досок, лежащие на этом столбце. На каждом шаге у нас какие-то доски будут присутствовать полностью, какие-то — частично (в этом случае они все еще будут прямоугольниками, пусть и неправильного размера), остальных досок не будет вовсе. Будем также на ходу перестраивать граф, но чуть по другому: разрешим ход конем на 2 клетки по вертикали и 0 по горизонтали; это не поменяет конечный граф, но поможет строить его на ходу. Количество вертикальных ребер может увеличиться только в том случае, если после добавления столбца появилась новая доска, которой раньше не было. Причем каждая новая доска добавляет не более 2 ребер. Таким образом, всего вертикальных ребер будет не более $2N$.



Теперь нужно построить граф быстрее, чем за $O(N^2)$. У каждой доски возьмем только 4 угловые клетки, затем для каждого столбца и для каждой строки составим список, где будут лежать угловые клетки, лежащие в этом столбце или строке, по порядку. Чтобы найти все вертикальные ребра, исходящие из какой-то одной доски, посмотрим на строку на 2 выше самой верхней строки этой доски. Бинарным поиском найдем самую левую доску, имеющую угловую клетку в этой строке, в которую можно сходить, и самую правую такую доску. Во все доски между ними тоже можно сделать ход. Таким образом мы найдем все доски, в которые можно попасть из текущей шагом на 2 клетки вверх и 1 клетку вбок. Аналогично разберем три других хода: 2 клетки вниз и 1 вбок, 2 клетки влево и 1 вверх или вниз, 2 клетки вправо и 1 вверх или вниз.

Явно построив граф, запустим на нем поиск в глубину. Сложности такого решения — $O(N \cdot log(N))$, так как для каждой доски мы выполняем бинарный поиск. На все остальное уходит всего $O(N)$ действий.

Подзадача 5: N ≤ 50000, M ≤ 10^9

Для этой подзадачи реализуем сжатие координат. Всего имеется не более 2N строк и не более 2N столбцов, в которых есть угловая клетка. Значит, можно хранить не более 4N списков и делать бинарный поиск только в том случае, если нужная строка или столбец присутствует. Сложность остается такой же — $O(N \cdot log(N))$.

Задача C: Правильные печеньки

В окружность вписаны N правильных многоугольников. Требуется определить, сколько из них хотя бы частично не покрыты другими. Число вершин многоугольника будем называть его степенью.

Подзадача 1: N ≤ 2

Для N = 1 ответ 1. Иначе из двух многоугольников один покрывает другой, только если его степень кратна степени другого. В этом случает ответ 1, иначе 2.

Подзадача 2: N ≤ 1000

Одинаковые многоугольники друг друга покрывают, поэтому из них видно не более одного (если он не покрыт многоугольником другой формы). Будем рассматривать только различные многоугольники из набора. Пользуясь свойством из решения подзадачи 1, получаем, что ответ — это число многоугольников, чьи степени не делят степень никакого другого многоугольника из набора. В данной подзадаче можно для каждого многоугольника проверить это свойство, проверив делимость его степени на степени каждого из остальных. Сложность решения $O(N^2)$.

Подзадача 3: N ≤ 50000

Известно, что степени многоугольников не превосходят 100000. Заведем массив флагов, в котором флаг по индексу k будет означать, что в наборе нет многоугольника, который бы перекрыл k-угольник. Инициализируем флаги значением ИСТИНА.

Делители степени a_i произвольного многоугольника можно перебрать как числа вида i и a_i/i для всех i от 2 до $\lfloor\sqrt{a_i}\rfloor$ (где $[x]$ — целая часть x). Таким перебором для каждого многоугольника из набора отметим в массиве флагов значением ЛОЖЬ все степени многоугольников, которые в данном наборе можно перекрыть. После этого ответ находится подсчетом всех уникальных многоугольников, для степени которых флаг установлен в значение ИСТИНА. Сложность такого решения составляет $O(N \cdot \sqrt{M})$, где M — максимальная возможная по условию задачи степень многоугольника.

Альтернативное решение: Для каждого числа от 3 до M заранее запишем, есть ли многоугольник с таким количеством вершин. После чего для каждого многоугольника определим, делит ли его количество вершин количество вершин другого многоугольника. Если у текущего многоугольника a_i вершин, то многоугольник, покрывающий его, имеет $k \cdot a_i$ вершин, где k — натуральное число, большее 1. Переберем все k от 2 до $M \ div \ a_i$ и для каждого посмотрим в заранее

составленном массиве, есть ли многоугольник с $k \cdot a_i$ вершинами. При этом важно рассматривать только многоугольники с различным числом вершин; если есть несколько одинаковых многоугольников, из них нужно рассмотреть только один.

Посчитаем время работы этого решения. Если все a_i различны, время работы составляет $O(M/a_1 + M/a_2 + \ldots + M/a_n)$. Это принимает наибольшее значение при $a_1 = 3,\ a_2 = 4,\ a_3 = 5,\ \ldots,\ a_n = n + 2$, тогда сложность работы решения составит $O(M/3 + M/4 + M/5 + \ldots + M/(N + 2))$. Можно доказать, что эта сумма имеет порядок $O(M \cdot log(N))$.

Задача D: Киберспортивный турнир

Игроки входят в игру и выходят из игры, а также обмениваются сообщениями: публичными и приватными. Каждое сообщение влияет на удачу игроков, необходимо уметь сравнивать значения удачи игроков.

Подзадача 1

В этой подзадаче нужно аккуратно реализовать все, что просят, кроме команды /quit, не опечататься в сообщениях и не допустить ошибок в реализации команд. Так как $A = B$, то можно считать, что мы не умножаем и делим удачу, а прибавляем и вычитаем ее; при этом нужно не забыть, что при $A = B = 1$ удачи всегда будут равны. Для поиска игроков

Подзадача 2

В этой подзадаче нужно добавить команду /quit и написать аккуратное сравнение рациональных чисел. Те, кто пишет на питоне, могут воспользоваться модулем fractions. В любом случае, можно хранить их в обычном типе вещественных чисел, так как ограничения небольшие.

Подзадача 3

Эта подзадача во многом сложнее предыдущей, кроме того, что здесь $A = B$. Это упрощает подсчет удачи: можно считать, что “gl” увеличивает удачу на 1, а “hf” уменьшает удачу на 1. Опять же, если $A = B = 1$, удачи всегда будут равны.

Подзадача 4

В этой подзадаче числа могут быть слишком большими, чтобы хранить их в обычном типе чисел с плавающей запятой. Чтобы считать удачу, воспользуемся следующим трюком: будем хранить не само значение удачи, а его логарифм. Логарифм — возрастающая функция, поэтому он сохраняет сравнение. Более того, если значение удачи равно A^x/B^y, то его логарифм равен $log(A) \cdot x - log(B) \cdot y$ (независимо от

основания логарифма). Таким образом, можно не умножать на A и делить на B, а прибавлять $log(A)$ и вычитать $log(B)$. Сравнивать числа нужно с эпсилоном, меньшим $log(1.0001)$, то есть, нужно зафиксировать константу $\varepsilon < log(1.0001)$ и считать два числа равными, если они отличаются не более, чем на ε.

Удобно хранить числа не в виде одного вещественного числа, а в виде показателей степеней при A и B. То есть, если число равно A^x/B^y, можно явно хранить x и y. Это позволит написать решение без эпсилона: числа в таком виде можно проверять на равенство точно, а если они не равны, сравнить значения $log(A) \cdot x - log(B) \cdot y$.

Подзадача 5

Основная сложность этой подзадачи — эффективная реализация команды /say. Так как в игре могут быть $O(N)$ игроков и может быть отправлено $O(N)$ сообщений /say, выполнение всех команд вручную займет $O(N^2)$ времени. Поймем, как делать это быстрее.

Для начала заметим, что, так как нам нужно только сравнивать числа, то можно уменьшать удачу всех игроков на одно и то же число, и это не на что не повлияет. Значит, при выполнении команды /say увеличить удачу всех игроков в игре, кроме себя — это все равно, что уменьшить удачу себя и всех игроков не в игре.

Следующий шаг — воспользоваться тем, что игроки, которые сейчас не в игре, ни с кем не сравниваются. Это значит, что необязательно поддерживать актуальную удачу игроков, которые не в игре, а можно пересчитать ее в тот момент, когда игрок входит в игру.

Сделать это можно следующим образом: будем сохранять все команды /say в хронологическом порядке. Для каждого игрока, который вышел из игры, запомним, в какой момент времени он вышел. Когда игрок возвращается обратно в игру, нужно уменьшить его удачу на сумму всех изменений команд /say, произошедших, пока он был вне игры. Это можно сделать с помощью массива префиксных сумм, насчитывая его на ходу; после очередной команды /say добавим к массиву префиксных сумм его последний элемент (изначально 0), увеличенный на изменение удачи текущим сообщением.

Подведем итог: команды /write и /compare выполняются точно так же, обращаясь к удаче игрока напрямую. Команда /say вместо того, чтобы увеличить удачу всех игроков в игре, кроме автора, уменьшит удачу автора напрямую, а также будет внесена в массив префиксных сумм. При выполнении команды /quit игрок будет запоминать текущую длину массива префиксных сумм, при повторном выполнении команды /join игрок вычтет из своей удачи сумму всех новых изменений, добавленных с тех пор, как он в последний раз вышел. При первом выполнении команды /join нужно вычесть из начальной удачи сумму всех изменений, то есть последний элемент массива префиксных сумм.

Решение работает за $O(N \cdot log(N))$ или за $O(N)$ в зависимости от того, какой структурой данных ищутся игроки по никам.

Задача E: Городки и роботы

На плоскости нарисован прямоугольник и отмечено N пар точек. Необходимо соединить парные точки кривыми так, чтобы кривые не пересекались между собой и не пересекали прямоугольник кроме концевых точек.

Подзадача 1: N = 1

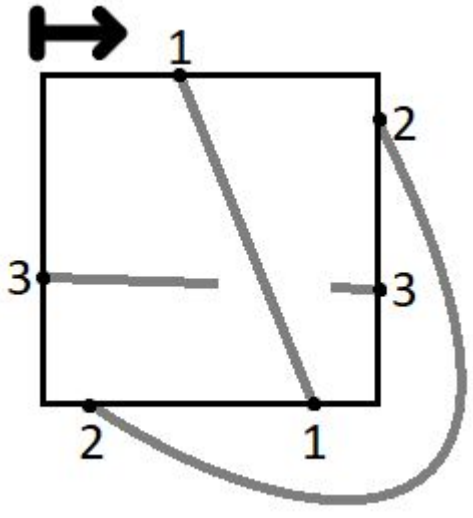
Есть всего одна пара точек. Если одна точка лежит строго внутри прямоугольника, а другая — строго снаружи него, то ответ NO, иначе — ответ YES.

Подзадача 2: N ≤ 2

Выполнить ту же проверку для каждой пары. Если для одной из пар одна точка лежит строго внутри прямоугольника, а другая — строго снаружи него, то ответ NO, иначе — ответ YES. Две кривые всегда можно нарисовать так, чтобы они не пересекались.

Подзадача 3: N ≤ 3

Кроме вышеуказанной проверки, нужно выполнить еще одну. Если есть 3 пары точек и каждая точка лежит на прямоугольнике, пойдем по кругу вдоль прямоугольника и запишем, какой паре принадлежит каждая точка. Если записано 123123 или аналогичное, то есть, каждая точка лежит между двумя точками из других пар, то нельзя соединить все пары кривыми без пересечений (задача «Три домика, три колодца»).



Подзадача 4: N ≤ 500

Обобщим предыдущие решения. Если есть пара точек такая, что одна лежит строго внутри прямоугольника, а другая — строго вне, то ответ NO.

Далее, заметим, что пары точек, в которых одна из точек лежит строго внутри или строго снаружи прямоугольника, на ответ не влияют. Можно провести кривые внутри этих пар самыми первыми, тогда каждая кривая не будет влиять на то, какие точки мы можем соединить.

Остались пары точек такие, в которых обе точки лежат на прямоугольнике. Если мы можем соединить точки внутри каждой из пар без пересечений, то каждая кривая будет лежать либо полностью внутри прямоугольника, либо полностью снаружи. Причем если две пары такие, что кривые внутри них могут пересечься, то их нужно пустить по разные стороны: одну кривую — внутри, другую снаружи.

Пойдём вдоль прямоугольника по кругу и для каждой встреченной точки запишем, в какой момент времени мы ее встретили. Тогда каждой интересующей нас паре точек будет сопоставлено два числа: время встречи первой и второй. Назовем их L_i и R_i, $L_i < R_i$. Тогда если найдутся две пары (L_i, R_i) и (L_j, R_j) такие, что $L_i < L_j < R_i < R_j$, то эти пары нужно пустить по разные стороны.

Теперь построим граф, вершинами которого будут пары точек. Между вершинами номер i и номер j проведем ребро в том и только в том случае, когда $L_i < L_j < R_i < R_j$ или $L_j < L_i < R_j < R_i$.

Осталось проверить граф на двудольность. Если граф двудолен, то ответ — YES; одна доля будет содержать все такие пары точек, которые мы соединим кривыми через внутреннюю часть прямоугольника, а другая доля — пары точек, которые мы соединим через наружную часть. Если граф не двудолен, ответ — NO.

Асимптотика такого решения — $O(N^2)$ из-за того, что количество ребер в таком графе может быть равно N^2.

Подзадача 5: N ≤ 100000

Сформулируем задачу по-другому. Будем идти вдоль прямоугольника по кругу. Для каждой пары точек скажем, что первая точка «открывает» ее, вторая — «закрывает» (первая и вторая в порядке обхода). Теперь наша задача — разбить все пары на две правильные скобочные последовательности.

Сделаем это следующим образом: будем поддерживать стек из «открытых» пар. Когда мы встретим первую точку из пары, добавим ее в стек. Когда мы встретим вторую точку из пары, если первая все еще в стеке, будем удалять точки с вершины стека, пока не найдем первую точку из этой же пары.

После завершения этого процесса мы будем знать, что все точки, которые мы не удалили из стека, можно соединить во внутренней части прямоугольника без

пересечений. Осталось проверить, что все удаленные точки можно соединить во внешней части без пересечений; сделать это можно еще одним проходом со стеком.

Асимптотика такого решения — $O(N \cdot log(N))$ из-за сортировки, необходимой, чтобы рассмотреть точки в порядке обхода.

Альтернативное решение: Вернемся к решению задачи через графы. Во-первых, заметим, что если мы получим разбиение графа на две доли, мы можем проверить это разбиение на корректность за $O(N \cdot log(N))$. Во-вторых, заметим, что если заменить граф остовным лесом, у него будет ровно одно корректное разбиение на две доли с точностью до инвертирования компонент связности.

Будем строить остовный лес с помощью структуры данных “система непересекающихся множеств”. Точно так же, как и в первом решении, будем идти по точкам в порядка обхода прямоугольника, но теперь будем поддерживать упорядоченное множество открытых точек вместо стека. Каждый раз, когда точка «закрывает» свою пару, найдем все пары из других компонент, пересекающиеся с текущей. Для этого нужно выполнить запрос “На отрезке множества найти точку из компоненты, отличной от данной”, после чего слить компоненты текущей и найденной точек и повторять запрос до тех пор, пока он что-то находит.

Чтобы выполнять такой запрос, будем поддерживать второе упорядоченное множество — множество отрезков точек из первого множества, лежащих в одной компоненте. Это делается аккуратными операциями с множеством; чтобы поменять компоненту точки, удалим ее и добавим заново с другой компонентой.

Поскольку вы меняем компоненты сразу нескольких точек при слиянии компонент, удобно это делать системой непересекающихся множеств с одной эвристикой без сжатия путей. Таким образом, СНМ выполняет $O(N \cdot log(N))$ действий, каждое из которых влечет операции над упорядоченным множеством, поэтому решение работает за $O(N \cdot log^2(N))$.

Задача F. Web-котики

В этой задаче будет показан самый простой путь решения по мнению автора без рассмотрения всевозможных случаев, и ещё одно альтернативное решение.

Для начала поймем что $W = (A + k) \cdot N + B \cdot (N + 1)$, где k — это целочисленное отклонение ширины картинки от того что хочет заказчик, а B — ширина отступа.

Выразим ширину из этой формулы: $B = (W - (A + k) \cdot N) / (N + 1)$. Так как в задаче все ширины (и картинок, и отступов) целочисленны, то рассмотрим, в каком случае B является целым числом. Немного преобразуем нашу формулу:
$B = (W - (A + k) \cdot N) / (N + 1) = (W + (A + k) - (A + k) \cdot (N + 1))) / (N + 1)$. Отсюда мы видим, что B целочисленно тогда и только тогда, когда $W + (A + k)$ делится на $N + 1$ без остатка.

Определим множество возможных значений для $A + k$ таких, что при них значение B — целое.

Таким множеством будет
$A + k \, \varepsilon \, \{N + 1 - W \, mod \, (N + 1) + (N + 1) \cdot t \mid t - \text{целое число}\}$. ***(1)***

В этом множестве присутствуют значения, применяя которые получаются некорректные значения в рамках нашей задачи, такие как нулевые или отрицательные значения ширины картинок/отступов.

Заметим что минимальное возможное валидное значение $A + k$ достигается при $t = 0$.

Найдем ближайшие к A значения $A + k$: Из формулы
$A + k = N + 1 - W \, mod \, (N + 1) + (N + 1) \cdot t$ выведем t:
$t = ((A + k) - (N + 1 - W \, mod \, (N + 1)) / (N + 1)$.

Вычислим это значение подставив вместо $k = 0$ и округлим по стандартным математическим правилам. Мы получили такое целое значение t, при котором, при подстановке в формулу (1) получается ближайшее к значению заказчика значение ширины картинок, при котором ширина отступов тоже целая.

Теперь нам необходимо найти такое максимальное число $A + k$ между значениями $t = 0$ и найденном t, которое не порождает невалидных значений ширины картинок и отступов, что и является ответом. С этой задачей прекрасно справляется бинарный поиск.

Если же ни при одном значении $A + k$ из данного промежутка для t, значения ширины картинок и отступов не становятся валидными, то ответ “NO”.

Альтернативное решение: Выразим ширину из формулы: $B = (W - (A + k) \cdot N) / (N + 1)$. Немного преобразуем эту формулу:

$$B = (W - (A + k) \cdot N) / (N + 1)$$

$$B = (W + (A + k) - (A + k) \cdot (N + 1))) / (N + 1)$$

$$B = (W + A + k) / (N + 1) - A - k$$

Отсюда мы видим, что B целочисленно тогда и только тогда, когда $W + A + k$ делится на $N + 1$ без остатка.

Не все k дают корректные решения; надо, чтобы ширина картинок и ширина отступов была положительна. Это дает нам условия $A + k \geq 1$ и $B \geq 1$. Преобразуя второе неравенство, получим

$$(W + A + k) / (N + 1) - A - k \geq 1$$

$$W + A + k - (A + k) \cdot (N + 1) \geq N + 1$$

$$W - AN \geq N + 1 + kN$$

$k \le (W - AN - N - 1)/N$

Так как k целое, можно даже сказать $k \le [(W - AN - N - 1)/N]$ (где $[x]$ — целая часть x).

Таким образом, нам нужно найти число k такое, что $l \le k \le r$, где $l = 1 - A$ и $r = [(W - AN - N - 1)/N]$, и $W + A + k$ делится на $N + 1$ без остатка, то есть $k \equiv - A - W \ (mod\ (N + 1))$. Из всех таких k нам нужно самое близкое к нулю, если таких несколько — положительное.

Очевидными кандидатами являются $(- A - W)\ mod\ (N + 1)$ и $((- A - W)\ mod\ (N + 1)) - N - 1$, самые близкие к нулю числа с нужным остатком (будем считать, что $x\ mod\ y$ всегда лежит на отрезке от 0 до $y - 1$). Эти два числа могут не входить в отрезок от l до r, в этом случае ответ будет рядом с концом отрезка. Ближайшее к левому концу отрезка число — или $l - ((l + A + W)\ mod\ (N + 1))$, или $l - ((l + A + W)\ mod\ (N + 1)) + N + 1$. Ближайшее к правому концу отрезка число — $r - ((r + A + W)\ mod\ (N + 1))$.

Итак, у нас есть 5 кандидатов:

- $k = (- A - W)\ mod\ (N + 1)$
- $k = ((- A - W)\ mod\ (N + 1)) - N - 1$
- $k = l - ((l + A + W)\ mod\ (N + 1))$
- $k = l - ((l + A + W)\ mod\ (N + 1)) + N + 1$
- $k = r - ((r + A + W)\ mod\ (N + 1))$

Нужно взять значения k, лежащие на отрезка от l до r, из них — ближнее к нулю, если таких несколько — положительное. После чего вывести $A + k$. Если подходящего значения среди кандидатов вообще нет, вывести “NO”.

Задача G: Любимые бутерброды

Для одного бутерброда нужно A хлеба и B колбасы. Когда Вале не хватает ингредиентов, он идет в магазин и покупает наименьшее количество хлеба и колбасы, чтобы ему хватило на один бутерброд, но хлеб продается батонами веса X, а колбаса — палками веса Y. Сколько раз Валя ходил в магазин, если он съел N бутербродов?

Подзадача 1: A, B, X, Y, N ≤ 100

Для начала, если $A \ge X$ или $B \ge Y$, то после каждого бутерброда нам нужно будет идти в магазин, и ответ будет равен N. Все последующие решения предполагают, что $A < X$ и $B < Y$.

Реализуем то, что от нас просят в условии. Запустим цикл из N итераций и будем поддерживать текущее количество хлеба и колбасы. Если хлеба меньше A или

колбасы меньше B, будем покупать хлеб, пока его не станет хотя бы A, и колбасу, пока ее не станет хотя бы B. За каждую такую покупку увеличим ответ на 1.

Подзадача 2: A, B, X, Y ≤ 100, N ≤ 10^9

Пусть мы съели $X \cdot Y$ бутербродов. Тогда мы съели $A \cdot X \cdot Y$ хлеба и $B \cdot X \cdot Y$ колбасы. Так как мы покупаем хлеб в количествах, кратных X, а колбасу — в количествах, кратных Y, то после съедения $X \cdot Y$ бутербродов у нас останется ровно 0 хлеба и колбасы. Если N больше, чем $X \cdot Y$, посчитаем циклом, сколько раз мы сходили в магазин, чтобы съесть $X \cdot Y$ колбасы, после чего мы сможем есть бутерброды пачками по $X \cdot Y$ штук, для каждой увеличивая ответ на одно и то же число. После чего останется $N \ mod \ (X \cdot Y)$ несъеденных бутербродов, которые мы сможем съесть также циклом. Сложность такого решения — $O(X \cdot Y)$.

Альтернативное решение: будем поддерживать состояние — количество хлеба и колбасы в запасе. Изначально у нас 0 хлеба и 0 колбасы, перед походом в магазин или хлеба меньше A, или колбасы меньше B, а после похода в магазин — хлеба не более $A + X$, колбасы не более $B + Y$. Таким образом, возможных состояний не более $(A + X) \cdot (B + Y)$. Будем есть бутерброды простым циклом, запоминая, сколько походов в магазин было сделано до каждого из состояний, пока не вернемся в состояние, где мы уже были. Зная длину цикла и количество походов в магазин за этот цикл, мы сможем есть бутерброды пачками, пока не останется съесть меньше, чем длина цикла, что в свою очередь меньше $(A + X) \cdot (B + Y)$. Сложность такого решения — $O((A + X) \cdot (B + Y))$.

Подзадача 3: A, B ≤ 100, X, Y, N ≤ 10^9

Усовершенствуем решение для предыдущей подзадачи. А именно, будем поддерживать количество хлеба k и количество колбасы t. Чтобы ускорить предыдущее решение, будем “перепрыгивать” между состояниями, съедая несколько бутербродов за раз и считая при этом число походов в магазин.

Первая оптимизация, которую мы применим, заключается в следующем: если у нас хватает ингредиентов на несколько бутербродов, съедим их все вместо того, чтобы есть по одному. Из этого получится, что на каждом шаге у нас либо хлеба меньше, чем A, либо колбасы меньше, чем B. Это уменьшает число состояний: на каждом шаге либо $t < A$ и $k < B + Y$, либо $t < A + X$ и $k < B$, и всего возможных состояний меньше, чем $A \cdot (B + Y) + (A + X) \cdot B$.

Этого недостаточно, может возникнуть случай, когда одного из ингредиентов слишком много. Для этого нужна вторая оптимизация. На каждом шаге посмотрим на ингредиент, которого больше, и посчитаем, сколько бутербродов нужно съесть до того, как придется покупать этот ингредиент. До покупки этого ингредиента нам придется ходить в магазин только за другим ингредиентом, и мы можем быстро посчитать, сколько походов в магазин это займет.

Например, если у нас имеется t хлеба и k колбасы, то до следующей покупки хлеба мы съедим $t\ div\ A$ бутербродов. За это время мы съедим $B \cdot (t\ div\ A)$ колбасы и сходим в магазин m раз, где m — наименьшее целое число, подходящее под условие $k - B \cdot (t\ div\ A) + m \cdot Y \geq 0$, что равносильно $m \geq (B \cdot (t\ div\ A) - k) / Y$. Отсюда можно найти число походов в магазин m.

Это уже дает нам решение, которое с одной стороны работает не более $A \cdot (B + Y) + (A + X) \cdot B \leq 2AB + 2(A + B) \cdot max(X, Y)$ шагов, и с другой стороны ест $max(X\ div\ A, Y\ \ div\ B) \geq max(X\ div\ max(A, B), Y\ \ div\ max(A, B)) = max(X, Y)\ div\ max(A, B)$ бутербродов за раз, что закончит выполняться за $O(N \cdot max(A, B) / max(X, Y))$ шагов. Таким образом, решение отработает за $O(min(2AB + 2(A + B) \cdot max(X, Y),\ N \cdot max(A, B) / max(X, Y)))$ шагов. Максимум достигается при наибольших A, B и N и таком $max(X, Y)$, что левая часть равна правой. Это дает асимптотику решения примерно $O(\sqrt{N} \cdot max(A, B))$.

Вузовско-академическая олимпиада по информатике

2019-2020 учебный год

Критерии определения призеров и победителей

Критерии определения призеров и победителей заключительного этапа 2019-2020 учебного года

Согласно решению жюри Вузовско-академической олимпиады по информатике, были установлены следующие критерии для присуждения дипломов I, II, III степеней:

Степень диплома	**Разбалловка**	**Участников**
		120
5-11 классы	Максимум 700 баллов	
I	От 400 баллов и выше	9
II	От 330 до 399 баллов включительно	9
III	От 280 до 329 баллов включительно	12
Количество призеров и победителей заключительного этапа		30
Процент призеров и победителей от общего числа участников заключительного этапа		25%