Вузовско-академическая олимпиада
по программированию на Урале

2020-2021 учебный год

Задания первого (отборочного) этапа

Задача A. ИИ

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

Работа в Яндексе шла своим чередом. Закончив работу над проектом, Стёпа и Катя решили продолжить создание нового искусственного интеллекта. Спустя множество проб и ошибок, они были близки к завершению. Всё, что осталось — калибровка. Процесс делится на три шага:

1. ИИ выдает два числа A и B.
2. На основе полученных чисел начинаются расчеты.
3. Проверяется контрольное значение.

Ребята решили не терять времени даром и запустили процесс. Но вот незадача, во время второго шага настало время обеда. Прием пищи — важная составляющая, а значит, его пропускать нельзя. Тогда Стёпа решил обратиться к Вам за помощью. Всё, что требуется — посчитать контрольное значение.

По словам Кати, алгоритм получения значения полностью зависит от числа B.

- В случае, если B неотрицательно, то необходимо к нулю B раз прибавить A;
- В противном случае, необходимо из нуля $|B|$ раз вычесть A.

Формат входных данных

В первой строке дано целое число A ($-10^9 \leqslant A \leqslant 10^9$).
Во второй строке дано целое число B ($-10^9 \leqslant B \leqslant 10^9$).

Формат выходных данных

Выведите единственное целое число — контрольное значение, необходимое для завершения процесса.

Система оценки

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	50	$0 \leqslant A \leqslant 10^4; B = 0$	
2	20	$0 \leqslant A, B \leqslant 10^4$	1
3	20	$-10^4 \leqslant A, B \leqslant 10^4$	1, 2
4	10	$-10^9 \leqslant A, B \leqslant 10^9$	1, 2, 3

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

Пример

стандартный ввод	стандартный вывод
3 4	12

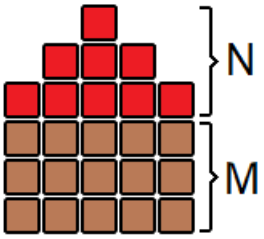
Задача B. Новый учебный корпус

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Программистов всем не хватает. Поэтому УрФУ увеличивает количество бюджетных мест на айтишных специальностях и открывает новые образовательные программы: `https://clck.ru/TU9iA`. Для комфортного обучения университет строит современный учебный корпус.

Архитекторы предложили проект будущего здания, поэтому пора оценить бюджет на строительство.

Здание можно условно разделить на две части: основание и крышу. Обе части состоят из квадратных панелей одинакового размера. Основание представляет собой прямоугольник высоты M, крыша представляет собой прямоугольный треугольник высоты N. Точную форму дома можно узнать по рисунку из проектной документации:



Ваша задача — посчитать точное количество панелей, которое необходимо для строительства.

Формат входных данных

В первой строке дано целое число M ($0 \leqslant M \leqslant 10^9$).
Во второй строке дано целое число N ($1 \leqslant N \leqslant 10^9$).

Формат выходных данных

Выведите единственное целое число — количество панелей, из которых будет состоять корпус.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	25	$M \leqslant 2, N \leqslant 2$	
2	25	$M = 0, N \leqslant 100$	
3	25	$M \leqslant 100, N \leqslant 100$	1, 2
4	25	$M \leqslant 10^9, N \leqslant 10^9$	1, 2, 3

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
2 2	10

Задача C. Простое сложение

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

Компания NAUMEN заботится не только о своих сотрудниках, но и о растущем поколении. Поскольку программист должен прекрасно работать с числами, в компании разрабатывается новая методика обучения детей, цель которой — научить детей работать с большими числами.

Прежде чем обучить сложению, предполагается обучить так называемому «простому сложению» (патент находится на рассмотрении). Простое сложение — это сложение двух чисел без переносов разряда. Например, если сложить числа 135 и 77 с помощью простого сложения, получится число 102. Кроме того, предлагается упростить процесс обучения ещё больше следующим образом: сначала научить складывать числа, состоящих из цифр от 0 до K. Так, если $K = 1$, будут складываться только числа, состоящие из нулей и единиц.

Конечно же, обучение будет автоматизировано с помощью проверяющей системы. Вам поручено задание написать программу, реализующую простое сложение чисел A и B.

Формат входных данных

В первой строке дано целое число K ($0 \leqslant K \leqslant 9$).

Во второй строке дано целое число A ($0 \leqslant A \leqslant 10^{1000}$).

Во третьей строке дано целое число B ($0 \leqslant B \leqslant 10^{1000}$).

Гарантируется, что числа A и B состоят только из цифр от 0 до K.

Формат выходных данных

Выведите единственное целое число — результат простого сложения чисел A и B без ведущих нулей (если сумма равна 0, выведите один ноль).

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$K \leqslant 4; A, B \leqslant 10^9$	
2	20	$A, B \leqslant 10$	
3	20	$A, B \leqslant 10^9$	1, 2
4	20	$A, B \leqslant 10^{18}$	1, 2, 3
5	20	$A, B \leqslant 10^{1000}$	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.**

Пример

стандартный ввод	стандартный вывод
4 1 3	4

Задача D. Привет, мир!

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

Первая программа, которую многие пишут, осваивая новый язык программирования — программа Hello, world! Однако, у Вас что-то пошло не так. Вместо того, чтобы вывести строку из 12 символов «Hello,world!», программа стала писать эту строку без остановки, даже не разделяя ни пробелами, ни переводами строк. Только когда было напечатано 10^9 символов, Вы нажали Ctrl+C и остановили программу.

Чтобы хоть как-то разобраться в этом хаосе, восстановите, что было выведено программой с L-го по R-й символ. Выведенные символы нумеруются, начиная с 1.

Формат входных данных

В первой строке дано целое число L.

Во второй строке дано целое число R ($1 \leqslant L \leqslant R \leqslant 10^9$, $R - L \leqslant 10^5$).

Формат выходных данных

В единственной строке выведите то, что написала программа с L-го по R-й символ. Вывод должен состоять только из символов, входящих в строку «Hello,world!» (строка не содержит пробела).

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	25	$R \leqslant 12$	
2	25	$R \leqslant 24$	1
3	25	$R \leqslant 10^5$	1, 2
4	25	$R \leqslant 10^9$	1, 2, 3

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
1 12	Hello,world!

Задача E. Игнат и игрушки

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

Игнат снова поссорился со своими родителями. Он решил собрать свои любимые игрушки в портфель и уйти жить к своему другу Роме.

К сожалению Игнат может носить портфель весом не более W, а это значит, что какие-то игрушки должны остаться у родителей. Чтобы меньше грустить, он для каждой игрушки определил её вес и радость, которую она ему приносит.

Помогите Игнату заполнить портфель игрушками так, чтобы максимизировать суммарную радость.

Формат входных данных

В первой строке заданы числа n, W ($1 \leqslant n \leqslant 10^5, 1 \leqslant W \leqslant 2 * 10^5$) — количество игрушек и максимальный вес, который может переносить Игнат.

Во второй строке заданы n чисел w_i ($1 \leqslant w_i \leqslant 2$) — вес i-й игрушки.

В третьей строке заданы n чисел c_i ($1 \leqslant c_i \leqslant 10^9$) — радость от i-й игрушки.

Формат выходных данных

Выведите единственное число — максимально возможную радость.

Система оценки

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	30	$n, W \leqslant 1000, w_i = 1, c_i \leqslant 1000$	
2	30	$n \leqslant 1000, W \leqslant 2000, c_i \leqslant 1000$	1
3	30	$n \leqslant 10^5, W \leqslant 2 * 10^5, c_i \leqslant 10^4$	1, 2
4	10	$n \leqslant 10^5, W \leqslant 2 * 10^5, c_i \leqslant 10^9$	1, 2, 3

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
3 3 2 1 2 1 3 2	5

Задача F. Транспортировка рамок

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

На день математика и механика студенты ФИИТ решили украсить стены своего факультета фотографиями любимых преподавателей. Для этого нужно напечатать фотографии и изготовить рамки для них.

Рамки толщиной в 5 миллиметров с произвольным размером делают в багетной мастерской, от которой придётся доставить до матмеха целых n рамок! Хорошо, что студенты уже прошли курс «Основы проектной деятельности» и знают, что такое управление рисками! Поэтому о том, как они будут тащить эти рамки на матмех, они подумали заранее.

Их план заключается в том, чтобы заказать рамки таких размеров, чтобы их все можно было компактно вложить друг в друга для удобства перевозки. Или более формально — для любой пары рамок i и j, i входит в j или наоборот. Естественно, рамки при этом можно поворачивать на угол, кратный 90°.

Зная размеры рамок в сантиметрах, определите, получится ли вложить друг в друга все рамки описанным выше способом или нет.

Формат входных данных

В первой строке дано число n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количество рамок.

В следующих n строках даны целые числа x_i, y_i ($1 \leqslant x_i, y_i \leqslant 10^9$) — размеры рамок в сантиметрах.

Формат выходных данных

Выведите «YES», если вложить рамки описанным выше способом возможно, и «NO» в противном случае.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	25	$n \leqslant 1000, x_i = y_i$	
2	25	$n \leqslant 1000$	1
3	50	$n \leqslant 2 \cdot 10^5$	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
3 1 1 2 2 3 3	YES
2 1 1 1 2	NO

Задача G. Сортировка

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

В языке программирования «Ы++» есть переменные, условные операторы и даже массивы, но нет циклов. Чтобы обработать элементы массива, нужно обращаться к каждому по отдельности. А отсортировать массив — вообще кошмарная задача.

Допустим, вам нужно отсортировать массив `ar` длины n. Элементы этого массива пронумерованы от 0 до $n-1$, обратиться к i-му элементу массива можно так: `ar[i]`. Чтобы отсортировать массив, достаточно кода, состоящего из комбинаций следующих двух строчек:

- «`if(ar[i]<ar[j])`», где i и j — индексы элементов массива. Эта строчка описывает условие. Следующая команда выполнится только в том случае, если i-й элемент массива меньше j-го.
- «`swap(ar[i],ar[j]);`», где i и j — индексы элементов массива. Эта строчка меняет местами элементы массива на i-й и j-й позициях.

Вы давно уже просили разработчиков языка «Ы++» включить в стандартную библиотеку алгоритм сортировки. Пришло время доказать, что это возможно! Напишите программу, которая для произвольного n выведет код на языке «Ы++», состоящий только из двух вышеуказанных команд и сортирующий массив длины n.

Формат входных данных

В единственной строке дано число n — длина массива `ar`, который нужно отсортировать ($2 \leqslant n \leqslant 400$).

Формат выходных данных

Выведите программу на языке «Ы++», сортирующий массив `ar` длины n. Каждая строчка должна иметь не более одной команды, которая в точности соответствует одной из двух вышеуказанных команд, в которые вместо «`i`» и «`j`» вставлены числа. Нельзя менять регистр букв, изменять знак сравнения, обращаться к несуществующим элементам массива. Но можно вставлять пустые строки и пробелы, не разделяя одну команду на две строки и не деля слова и числа на части.

Получившаяся программа на языке «Ы++» должна иметь не более n^2 строк и занимать не более 4 мегабайт (4 194 304 символов).

Программа будет считаться корректной, если она корректно отсортирует 1000 случайно сгенерированных массивов длины n. Гарантируется, что для каждого n набор тестовых массивов одинаков для каждой посылки.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$n = 2$	
2	20	$n \leqslant 3$	1
3	20	$n \leqslant 4$	1, 2
4	20	$n \leqslant 40$	1, 2, 3
5	20	$n \leqslant 400$	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.**

Пример

стандартный ввод	стандартный вывод
2	if (ar[0] < ar[1]) swap(ar[0], ar[1]);

Замечание

Обратите внимание, что ответ в примере является неверным, однако он является корректной программой на языке «Ы++» и приведён только для демонстрации формата вывода.

Задача H. Сложность решётки

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

Всем студентам ФИИТ известно, что лучшие конспекты ведёт Паша Вилкин. Такой красоты Вы ещё не видели! Ходят даже слухи, что конспекты дают своему обладателю суперсилу — сдавать все экзамены на отлично! Естественно, каждый двоечник готов на всё, чтобы заполучить себе оригиналы.

Паша, как отличник, живёт в лучшем общежитии университета. Там есть всё, что нужно для комфортной жизни, даже комната для хранения драгоценных конспектов, с крепкой решётчатой дверью. Дверь состоит из металлических прутьев, представляющих из себя бесконечно длинные и бесконечно тонкие прямые линии.

Но наш герой не уверен в том, что однажды решётку не смогут распилить. Чтобы оценить вероятность взлома, Паша ввёл понятие «сложности решётки». Под ним он подразумевает количество различных пар прутьев, которые имеют хотя бы одну общую точку.

Найдите сложность решётки!

Формат входных данных

В первой строке дано целое число n — количество прутьев ($1 \leqslant n \leqslant 10^5$).

В каждой из следующих n строк заданы целые числа $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$ через пробел — координаты двух различных точек i-го прута ($-10^9 \leqslant x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2} \leqslant 10^9$).

Формат выходных данных

В единственной строке выведите сложность данной решётки — количество пар (i, j) таких, что $i < j$ и i-й прут имеет хотя бы одну общую точку с j-м.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$n \leqslant 10$, для любых 2 прямых угол наклона различный	
2	25	$n \leqslant 10^3$, для любых 2 прямых точек пересечения меньше 2	1
3	25	$n \leqslant 10^3$	1, 2
4	40	$n \leqslant 10^5$	1, 2, 3

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
3 1 1 2 1 1 1 2 2 1 1 1 2	3

Вузовско-академическая олимпиада
по программированию на Урале

2020-2021 учебный год

Задания второго (заключительного) этапа

Задача A. Снова ИИ

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

Стёпа и Катя работают в Яндексе и создают новый искусственный интеллект для научно-исследовательских целей. ИИ работает с целыми числами, особенно с числами от 1 до N. Раскрыть предназначение ИИ они пока не готовы, но известно, что для него важно свойство *обратимости* чисел: так они называют тот факт, что $N - (N - X) = X$ для всех X от 1 до N.

Вы решили на шаг опередить разработки Яндекса и создать аналогичный ИИ, который тоже будет работать с числами от 1 до N, но будет сосредоточен на операциях умножения и деления чисел. Для этого вы определили своё свойство *суперобратимости*. Будем говорить, что целое число X ($1 \leqslant X \leqslant N$) *суперобратимо* для N, если $N \ div \ (N \ div \ X) = X$, где div — операция целочисленного деления с округлением вниз.

К сожалению, не все числа оказались суперобратимы. Но нельзя терять время! Нужно скорее продолжить работу, а для этого нужно определить, сколько целых чисел от 1 до N являются суперобратимыми. Возможно, придётся даже перебрать несколько значений N_i, пока вы не найдёте подходящее.

Формат входных данных

В первой строке дано целое число Q — количество чисел N_i, которые вы планируете рассмотреть ($1 \leqslant Q \leqslant 10$).

Далее идут Q строк. В каждой строке дано целое число N_i ($1 \leqslant N_i \leqslant 10^{18}$).

Формат выходных данных

Выведите Q строк. В i-й строке должно быть записано единственное целое число — количество суперобратимых чисел для N_i.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	34	$N_i \leqslant 10^5$	
2	34	$N_i \leqslant 10^9$	1
3	32	$N_i \leqslant 10^{18}$	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 1 2 3 12	1 2 2 6

Задача B. Как пройти в столовую

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

Петя поступил на ФИИТ УрФУ и всё ещё не запомнил, как ориентироваться в учебном корпусе — матмехе. После второй пары Петя бесцельно слоняется по коридорам, которые представляют из себя сплошной лабиринт. Корпус имеет форму прямоугольника ширины W и высоты H, разделённого на клетки 1×1. Клетки бывают четырёх типов:

- Пустая клетка;
- Клетка со стеной;
- Клетка с ключом;
- Клетка с запертой дверью.

Как назло уборщицы регулярно начинают мыть пол, поэтому если Петя начнёт двигаться в одном направлении, то он не остановится, пока не упрётся в препятствие. Препятствиями являются граница корпуса, стена, а также запертая дверь, если у Пети нет ключа.

По пути Петя может собирать ключи. Проходя мимо ключа, Петя моментально подбирает его, не останавливаясь. Он может носить с собой неограниченное количество ключей. Любой ключ подходит к любой двери. Если Петя упирается в запертую дверь, он моментально открывает её ключом и продолжает движение без остановки. Ключ остается в замочной скважине двери. Если же у Пети нет с собой ни одного ключа, то запертые двери для него не отличаются от стен.

Также известно, что во время пар декан открывает все двери. Поэтому во время пар на матмехе нет ни одной запертой двери и ни одного ключа. Во время перемены можно найти ключи и/или запертые двери.

Помогите Пете не заблудиться и выбраться в столовую. Он знает, в какой клетке он начал свой путь, а также всю историю своих передвижений. Он сделал L шагов, каждый из шагов — перемещение в одном из четырех направлений: вверх, вправо, вниз или влево. Для каждого из шагов выведите, сколько клеток лабиринта он проскользил в этом направлении.

Формат входных данных

В первой строке даны три целых числа H, W и D — высота корпуса, ширина корпуса, а также число D, обозначающее время; если $D = 0$, то дело происходит во время пары, иначе — на перемене ($1 \leqslant H, W \leqslant 10^5$, $H \cdot W \leqslant 10^5$, $0 \leqslant D \leqslant 1$).

Далее идут H строк, описывающих корпус матмеха. Каждая строка состоит из W символов, не разделённых пробелами. Всего есть 5 возможных символов. Символ «.» (код 46) обозначает пустую клетку. Символ «#» (код 35) обозначает стену. Символ «K» обозначает клетку с ключом. Символ «L» обозначает клетку с запертой дверью. Символ «S» обозначает клетку, где начинает Петя — эта клетка считается пустой. Гарантируется, что есть ровно один символ «S».

Гарантируется, что если $D = 0$, то символы «K» и «L» в описании корпуса отсутствуют. Если $D = 1$, то есть хотя бы один символ «K» или «L».

В следующей строке дано целое число L — количество шагов, сделанных Петей ($1 \leqslant L \leqslant 10^5$).

В последней строке даны L символов, не разделённых пробелами. Всего есть 4 возможных символа, которые могут встречаться: «U», «R», «D», «L». Они обозначают направления вверх, вправо, вниз и влево соответственно. Корпус ориентирован так, как дан во вводе: первая строка является самой верхней, первый столбец является самым левым.

Формат выходных данных

Выведите L строк, в каждой строке выведите одно целое число. В i-й строке выведите количество клеток, которые Петя прошёл, сделав i-й шаг.

Система оценки

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	25	$H \cdot W \leqslant 100$, $L \leqslant 100$, $D = 0$	
2	25	$H \cdot W \leqslant 100$, $L \leqslant 100$, $D \leqslant 1$	1
3	25	$H \cdot W \leqslant 10^5$, $L \leqslant 10^5$, $D = 0$	1
4	25	$H \cdot W \leqslant 10^5$, $L \leqslant 10^5$, $D \leqslant 1$	1, 2, 3

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.**

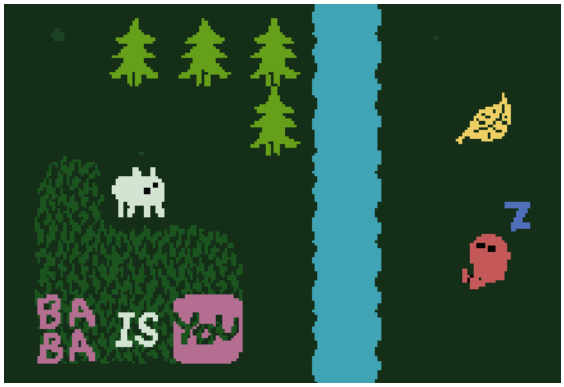
Примеры

стандартный ввод	стандартный вывод
3 4 0 S..# .... #... 6 RDRULD	2 2 1 1 3 0
1 6 1 LKSKLL 4 RRLR	2 0 4 4

Задача C. Baba is You

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

Вы отправились исследовать таинственный мир, заполненный различными существами и предметами. Поскольку многие объекты в этом мире не были открыты ранее, вы дали каждому объекту название. Все названия состоят из заглавных латинских букв и имеют длину от 1 до 4 символов.



В ходе наблюдений вы заметили, что объекты со временем меняются. Это вы решили записать в журнал с помощью *правил*. Всего получилось N правил. Правило номер i гласит, что объект A_i со временем превратится в объект B_i. Такое правило записывается как «A_i `IS` B_i». Заметим, что объекты A_i и B_i могут совпадать, в этом случае превращение всё равно происходит.

Сегодня вы вернулись в исследовательский центр и решили заняться теоретическими экспериментами. Вам дали план, состоящий из Q запросов. У вас нет при себе экземпляров изученных объектов, поэтому при выполнении запросов приходится опираться на записи в журнале. Каждый запрос задаётся одним из двух способов:

1. «`?` C_j». Чтобы выполнить этот запрос, нужно провести эксперимент и определить, во что в конечном итоге превратится объект C_j согласно записанным в журнале правилам.

2. «`+` C_j `IS` D_j». Чтобы выполнить этот запрос, нужно добавить новое правило в журнал, согласно которому объект C_j превращается в объект D_j. Новое правило не отличается от старых, и оно начинает работать для всех последующих запросов.

Важная часть любой исследовательской деятельности — ведение отчёта об ошибках. В ходе проведения эксперимента может произойти ошибка одного из двух типов:

- «`INFINITE LOOP`»: эта ошибка происходит в том случае, если цепочка превращений объекта бесконечна, то есть объект будет превращаться в другой объект сколько угодно раз. Ошибка не возникает, если объект превращается сам в себя.

- «`TOO COMPLEX`»: эта ошибка происходит в том случае, если согласно записанным правилам объект одновременно должен превратиться в два разных объекта. При возникновении этой ошибки эксперимент нужно тут же прекратить. Ошибка не возникает, если объект превращается в один и тот же объект по двум различным правилам. Но ошибка возникает, если объект должен превратиться в себя и в другой объект.

Попробуйте для каждого запроса первого вида определить, во что превратится данный объект или какая ошибка возникнет в результате выполнения запроса. Обратите внимание, что в одном запросе не могут произойти две ошибки сразу.

Формат входных данных

В первой строке дано целое число N — изначальное количество правил ($1 \leqslant N \leqslant 50000$).

Далее идут N строк. Строка номер i имеет вид «A_i IS B_i» (без кавычек) и описывает правило, согласно которому объект A_i превращается в объект B_i.

В следующей строке дано целое число Q — количество запросов ($1 \leqslant Q \leqslant 50000$).

Далее идут Q строк. Строка номер j имеет вид «? C_j» или «+ C_j IS D_j» (без кавычек) и описывает запрос первого или второго вида соответственно. Запросы выполняются в том порядке, в котором они даны во входных данных. Гарантируется, что существует хотя бы один запрос первого вида.

Все объекты названы строками длины от 1 до 4, состоящими из заглавных латинских букв. Гарантируется, что не существует объекта «IS». Также в описании правил встречается слово «IS», написанное заглавными латинскими буквами, а в описаниях запросов встречаются символы «?» (код 63) и «+» (код 43).

Формат выходных данных

Для каждого запроса первого вида, если ошибка не возникает, выведите в отдельной строке название объекта, в который превратится данный объект. Если ошибка возникает, выведите в отдельной строке «INFINITE LOOP» для ошибки первого типа или «TOO COMPLEX» для ошибки второго типа (в обоих случаях без кавычек).

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$N, Q \leqslant 50$, нет запросов на добавление, нет ошибок	
2	18	$N, Q \leqslant 50$	1
3	20	$N, Q \leqslant 1000$	1, 2
4	22	$N, Q \leqslant 50000$, нет ошибок	1
5	20	$N, Q \leqslant 50000$	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 необычные ограничения в подзадачах 1 и 4. В них гарантируется, что при выполнении превращений не возникает ошибок. В подзадаче 1 также гарантируется, что нет запросов второго вида.

Пример

стандартный ввод	стандартный вывод
3 BABA IS YOU BOX IS KEY KEY IS BOX 8 + BABA IS YOU ? BABA + YOU IS YOU ? BABA ? BOX + BOX IS BOX ? BOX ? LOVE	YOU YOU INFINITE LOOP TOO COMPLEX LOVE

Задача D. Он вам не ферзь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

В коридорах матмеха, где учатся студенты ФИИТ, есть много всего интересного: рояль на потолке, Хогвартс, оптические иллюзии на полу, качели, даже портал на платформу $9\frac{3}{4}$. А ещё есть шахматы на стене, в которые любят играть студенты Артемий и Михей. Во время каникул на матмехе начался ремонт, поэтому доску сняли.

Ребята не растерялись и решили нарисовать шахматы мелом на стене. За каникулы правила игры они подзабыли. Начертив мелом доску $M \times M$, они начали вспоминать, как должны быть расположены фигуры. Михей недолго думал и нарисовал на поле N белых королей — король ведь лучшая фигура! Артемий разбирался в шахматах побольше, и он знал, что лучшая фигура — это ферзь. Поэтому Артемий решил нарисовать на свободной клетке одного чёрного ферзя, который поставил бы под бой всех белых королей и при этом сам не был бы под боем.

По авторитетному мнению Михея, король может бить 8 смежных клеток, имеющих общую сторону или общий угол с той клеткой, на которой стоит сам король. Ферзь же полностью бьёт горизонталь, вертикали и обе диагонали, на которых стоит. При этом наши герои забыли, что ферзь не может прыгать через фигуры: по их мнению, если два короля стоят на одной линии и по одну сторону от ферзя, ферзь может бить их обоих.

Всего они провели T игр, каждый раз заново рисуя доску и фигуры. Зная, как Михей располагал своих королей, посчитайте количество позиций, куда Артемий мог поставить ферзя так, чтобы он не стоял под боем ни одного из королей и при этом все короли стояли под его боем.

Формат входных данных

В первой строке дано целое число T — количество игр ($1 \leqslant T \leqslant 10$).

Далее идут T блоков, каждый блок описывает очередную игру.

В первой строке блока даны два целых числа M и N — длина стороны поля и количество королей ($1 \leqslant M \leqslant 10^9$, $1 \leqslant N \leqslant 10^5$). Гарантируется, что Михей и Артемий сыграли не более одной игры с $N > 5$.

Далее идут N строк, в i-й из них записаны два целых числа x_i, y_i — номер столбца и номер строки, на которых стоит i-й белый король ($1 \leqslant x_i, y_i \leqslant M$). Гарантируется, что в каждой клетке стоит не более одного короля.

Формат выходных данных

Выведите T строк. В каждой строке выведите единственное целое число — количество позиций, куда Артемий может поставить чёрного ферзя в очередной игре.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	14	$M \leqslant 100$, $N = 1$	
2	19	$M, N \leqslant 100$	1
3	20	$M, N \leqslant 1000$	1, 2
4	28	$M, N \leqslant 10^5$	1, 2, 3
5	19	$M \leqslant 10^9$, $N \leqslant 10^5$	1, 2, 3, 4

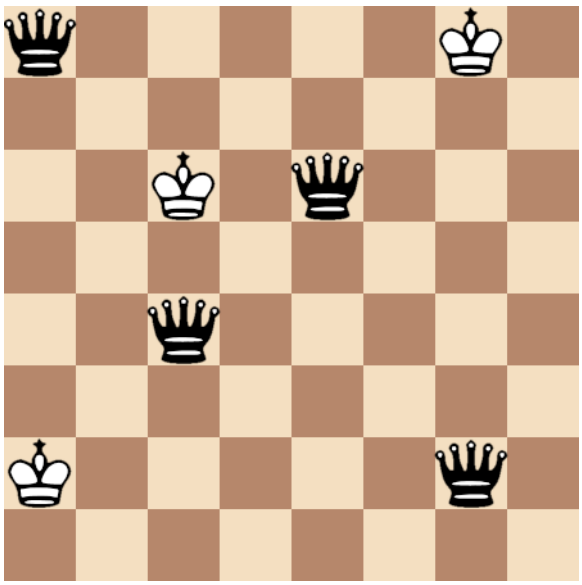
Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены, а также решение **корректно работает на примерах из условия.**

Пример

стандартный ввод	стандартный вывод
2 8 3 1 2 3 6 7 8 2 4 1 1 1 2 2 1 2 2	4 0

Замечание

Иллюстрация к первой игре из примера, одновременно показаны все возможные позиции ферзя:



Задача E. Сила есть, ум тоже нужен

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

В компании NAUMEN очень ценятся личные качества сотрудников. Основными личными качествами являются сила и ум. Недавно компания провела тестирование всех своих сотрудников, в результате которого были определены показатели силы и ума. По мнению исследователей, эти параметры позволят объективно оценить работоспособность каждого сотрудника.

Всего в компании N сотрудников, пронумерованных от 1 до N. Компания имеет иерархическую структуру. Чем меньше номер сотрудника, тем он главнее. Сотрудник номер 1 является директором, у него нет начальника. У всех остальных сотрудников есть начальник: у сотрудника номер i начальником является сотрудник номер P_i ($P_i < i$). Также принято считать, что $P_1 = 0$.

В результате исследования было определено, что i-й сотрудник имеет силу A_i и ум B_i. Эти показатели важны при решении проблем. Исследователи компании NAUMEN предложили следующую математическую модель проблемы: по их мнению, проблема — это пара натуральных чисел (x, y). Первое число описывает физическую сложность проблемы, второе число — умственную. Считается, что сотрудник номер i может самостоятельно решить проблему (x, y), если $A_i \geqslant x$ и $B_i \geqslant y$. Таким образом, учитываются как сила, так и ум.

Конечно, сотрудники не всегда решают проблемы поодиночке. Если у сотрудника в подчинении есть люди, он может делегировать любому своему непосредственному подчинённому эту проблему. Тот сотрудник, кому проблема была делегирована, может в свою очередь делегировать её одному из своих подчинённых, и так далее. Таким образом, если хотя бы один из подчинённых i-го сотрудника может решить проблему, то считается, что и i-й сотрудник может её решить.

Вам было поручено подведение итогов исследования. Для этого нужно для каждого сотрудника посчитать, сколько различных проблем он умеет решать. Обратите внимание, что раз параметры x и y в любой проблеме (x, y) — натуральные числа, то множество различных проблем, которые умеет решать сотрудник, всегда конечно.

Формат входных данных

В первой строке дано целое число N — количество сотрудников в компании NAUMEN ($1 \leqslant N \leqslant 50000$).

Далее идут N строк, описывающих сотрудников. В i-й строке даны три целых числа P_i, A_i, B_i — номер непосредственного начальника, показатель силы и показатель ума i-го сотрудника ($0 \leqslant P_i < i$, $1 \leqslant A_i, B_i \leqslant 10^6$; $P_i = 0$ только при $i = 1$).

Формат выходных данных

Выведите N строк. В i-й строке выведите одно целое число — количество различных проблем, которые умеет решать i-й сотрудник.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	21	$N \leqslant 5, A_i \leqslant 10, B_i \leqslant 10$	
2	26	$N \leqslant 50$, $A_i \leqslant 100$, $B_i \leqslant 10^6$	1
3	23	$N \leqslant 5000$, $A_i \leqslant 100$, $B_i \leqslant 10^6$	1, 2
4	14	$N \leqslant 500$, $A_i \leqslant 10^6$, $B_i \leqslant 10^6$	1, 2
5	16	$N \leqslant 50000$, $A_i \leqslant 10^6$, $B_i \leqslant 10^6$	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.**

Пример

стандартный ввод	стандартный вывод
3 0 5 5 1 1 10 1 4 3	30 10 12

Задача F. Трудный выбор

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

Ваня учится на ФИИТ в УрФУ, где каждый семестр студенты делают командные проекты. У Вани много друзей в группе, но у каждого свои требования. Например, если позвать в команду Петю, то должен прийти и Алёша, иначе Петя не придёт. Ваня по данным требованиям смог выяснить, что у него есть целых N вариантов, кого пригласить. Каждый из способов он описал в виде величины A_i — количество друзей в выборке под номером i.

Но это было полбеды! Ведь после того, как он с друзьями соберёт команду, нужно будет выбрать того, кто будет представлять проект на защите. Все ребята любят программировать, и никто не хочет заниматься презентацией, поэтому они договорились, что защищает проект тот, кого выберут по считалочке.

Ваня, будучи организатором этого выбора, должен проводить этот расчёт. Чтобы выбрать ответственного по считалочке, он выстраивает людей по кругу и начинает считать от себя, считая по одному слову на человека. На кого попадёт последнее слово, тот и защищает проект. Иван знает Q считалочек. Считалочка номер j состоит из B_j слов. Он решил узнать, насколько удачной будет каждая из них. Для этого он ввёл понятие *хорошей* компании одногруппников — таковой она является, если в ней он не будет защищать проект при использовании данной считалочки.

Ваня хочет уже начать писать код для проекта вместо поиска идеальной считалочки, поэтому он обратился к вам, чтобы вы помогли ему вычислить ответ. Ему нужно для каждой считалочки найти минимальный номер *хорошей* группы.

Формат входных данных

В первой строке задано целое число N — количество групп из друзей Вани ($1 \leqslant N \leqslant 10^5$).

Во второй строке записаны N целых чисел A_i — размеры i-й группы ($1 \leqslant A_i \leqslant 10^7$).

В третьей строке дано целое число Q — количество считалочек ($1 \leqslant Q \leqslant 10^6$).

В следующих Q строках записано по одному целому числу B_j — количество слов в j-й считалочке ($1 \leqslant B_j \leqslant 10^7$).

Формат выходных данных

Выведите Q строк, где j-я строка содержит одно целое число — минимальный номер *хорошей* группы для считалочки под номером j или -1, если таковой нет.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$N, Q \leqslant 100$; $A_i, B_j \leqslant 10^3$	
2	30	$N, Q \leqslant 10^3$; $A_i, B_j \leqslant 10^6$	1
3	16	$N \leqslant 5000$, $Q \leqslant 10^5$; $A_i, B_j \leqslant 10^6$	1, 2
4	16	$N \leqslant 10^5$, $Q \leqslant 5000$; $A_i, B_j \leqslant 10^6$	1, 2
5	15	$N, Q \leqslant 10^5$; $A_i, B_j \leqslant 10^6$	1, 2, 3, 4
6	13	$N \leqslant 10^5$, $Q \leqslant 10^6$; $A_i, B_j \leqslant 10^7$	1, 2, 3, 4, 5

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
3 1 2 5 3 3 7 12	2 -1 1

Замечание

Объяснение примера:

Пусть первая считалочка будет «Выступать будешь ты».

Если мы возьмём первую группу, то Ваня и Вася будут стоять по кругу. Считалочка будет воспроизводиться следующим образом: («Выступать», Ваня), («будешь», Вася), («ты», Ваня). Т.е. Ваня будет выступать с докладом.

Если возьмём вторую группу, то у нас будет круг: Ваня, Валя, Витя. В таком случае будет следующая ситуация: («Выступать», Ваня), («будешь», Валя), («ты», Витя) — в этом случае выступает не Ваня, а значит группа *хорошая* и она будет ответом.

Для второй считалочки для всех трёх групп в конце будет Ваня, а значит ответ -1.

Для третьей считалочки первая группа уже *хорошая*.

Задача G. Ulearn

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

Паша Вилкин, как и остальные студенты ФИИТ, проходит много разных курсов на платформе Ulearn.me. В последнее время у него скопилось много задачек на программирование, которые нужно успеть решить к дедлайну. Чтобы написать качественные решения и уложиться в срок, нужно пройти 4 этапа:

1. Написать код;
2. Отправить код на проверку автотестами;
3. Внести правки по результатам тестирования;
4. Отправить решение на код-ревью.

Первый и третий этапы требуют присутствия Паши и занимают время A. Для второго и четвёртого этапов присутствие Паши не требуется, каждый из них занимает время B. Чтобы решение получилось качественным, весь процесс должен идти без остановки. Помогите Паше рассчитать минимальное время, необходимое для решения N задач.

Формат входных данных

В первой строке дано целое число N ($1 \leqslant N \leqslant 10^9$).
Во второй строке дано целое число A ($1 \leqslant A \leqslant 10^5$).
В третьей строке дано целое число B ($1 \leqslant B \leqslant 10^5$).

Формат выходных данных

Выведите единственное целое число — минимальное необходимое время для решения всех задач.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	17	$B < A, N \leqslant 10^5$	
2	15	$B \leqslant A, N \leqslant 10^5$	1
3	16	$B \leqslant 2A, N \leqslant 10^5$	1, 2
4	26	$N \leqslant 10^5$	1, 2, 3
5	26	$N \leqslant 10^9$	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
2 3 1	15
2 2 2	10

Замечание

Иллюстрация к первому примеру:

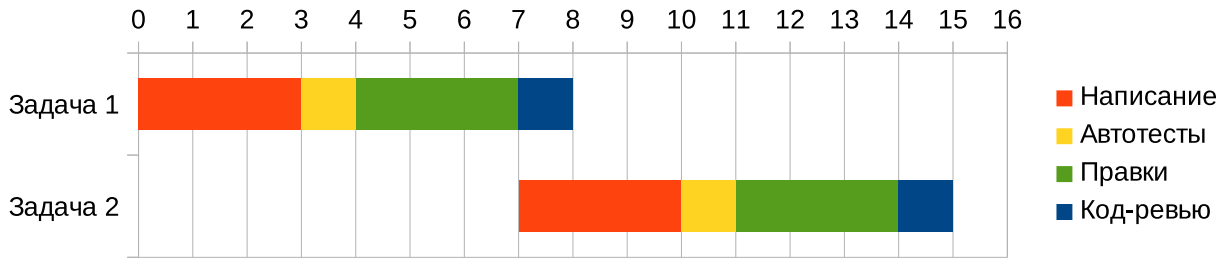


Иллюстрация ко второму примеру:

Вузовско-академическая олимпиада
по программированию на Урале

2020-2021 учебный год

Разбор заданий второго (заключительного) этапа

Разборы заданий с демонстрацией решений от жюри в формате видео доступны на YouTube: https://youtu.be/sFYzwsfk4Xk

Задача A. Снова ИИ

В первой подзадаче (**34 балла**) достаточно перебрать все числа от 1 до N и для каждого числа X проверить, верно ли, что $N \ div \ (N \ div \ X) = X$.

Пусть X супер-обратимо для N, и $Y = N \ div \ X$. Тогда выполняется $X = N \ div \ Y$. К тому же, Y супер-обратимо для N: если поделить N нацело на обе части равенства $X = N \ div \ Y$, получим $N \ div \ X = N \ div \ (N \ div \ Y)$, отсюда $Y = N \ div \ (N \ div \ Y)$.

Таким образом, все супер-обратимые числа образуют пары (X, Y) такие, что $N \ div \ X = Y$ и $N \ div \ Y = X$. Отсюда $X \cdot Y \leqslant N$. Значит, хотя бы одно число в паре не превосходит $\sqrt{N}$. Получаем решение для второй подзадачи (**34 балла**): перебираем все X от 1 до $\left\lfloor \sqrt{N} \right\rfloor$, проверяем каждое число X и каждое число $N \ div \ X$.

Чтобы решить третью подзадачу (**32 балла**), нужно более подробно посмотреть, какие числа образуют пары (X, Y). Оказывается, любое число $X \leqslant \left\lfloor \sqrt{N} \right\rfloor$ является супер-обратимым. Докажем это от противного.

1. Пусть $X \leqslant \left\lfloor \sqrt{N} \right\rfloor$ и $N \ div \ (N \ div \ X) < X$. Так как $N \ div \ X = Y$, выполняется $X \cdot Y \leqslant N$. Если $N \ div \ Y < X$, то $X \cdot Y > N$. Противоречие.
2. Пусть $X \leqslant \left\lfloor \sqrt{N} \right\rfloor$ и $N \ div \ (N \ div \ X) > X$. Так как $N \ div \ X = Y$, выполняется $X \cdot Y \leqslant N$ и $X \cdot (Y + 1) > N$. Если $N \ div \ Y > X$, то $Y \cdot (X + 1) \leqslant N$. Отсюда получаем, что $Y \cdot (X+1) < X \cdot (Y+1)$, следовательно, $Y < X$, или же $Y+1 \leqslant X$. Но тогда $X \cdot (Y+1) \leqslant X \cdot X \leqslant N$, это противоречит тому, что $X \cdot (Y + 1) > N$.

Таким образом, для $X \leqslant \left\lfloor \sqrt{N} \right\rfloor$ остаётся только один вариант — $N \ div \ (N \ div \ X) = X$. Получаем, что все числа $X \leqslant \left\lfloor \sqrt{N} \right\rfloor$ супер-обратимы для N.

Итак, существует $\left\lfloor \sqrt{N} \right\rfloor$ пар супер-обратимых чисел (X, Y). Ответ равняется $2 \cdot \left\lfloor \sqrt{N} \right\rfloor$, но из этого числа нужно вычесть единицу, если есть $N \ div \ \left\lfloor \sqrt{N} \right\rfloor = \left\lfloor \sqrt{N} \right\rfloor$, так как в этом случае есть пара из двух одинаковых элементов $(\left\lfloor \sqrt{N} \right\rfloor, \left\lfloor \sqrt{N} \right\rfloor)$.

Задача B. Как пройти в столовую

Для решения первой подзадачи (**25 баллов**) нужно сделать то, что просят — просто перемещаемся по лабиринту, аккуратно проверяя каждый шаг в лабиринте. Время работы составит $O(H \cdot W + L \cdot \max(H, W))$.

Решение второй подзадачи (**25 баллов**) также требует простого перемещения по лабиринту, однако теперь нужно дополнительно держать счётчик ключей, не забыть, что каждый ключ мы берём по одному разу, а каждую дверь открываем ровно один раз. Сложность решения аналогичная $O(H \cdot W + L \cdot \max(H, W))$.

В третьей подзадаче (**25 баллов**) обратите внимание, что поле не укладывается в статический массив. Нужно либо хранить массив строк динамической длины, либо записать поле в одну строку и аккуратно переводить координаты из двумерных в одномерные. Чтобы решить подзадачу, для каждой пустой клетки для каждого направления заранее посчитаем длину шага. Если считать в правильном направлении, это можно сделать за $O(1)$ для каждой клетки: ответ для клетки либо 0, если дальше по направлению идёт стена или граница поля; иначе — 1 плюс ответ для следующей клетки. Получаем время работы для подзадачи $O(H \cdot W + L)$.

Для решения четвёртой подзадачи (**25 баллов**) поймём, что перемещение по пустым клеткам бесполезно — нам важно лишь искать ближайшую из трёх важных клеток: стена, дверь, ключ (можно считать, что поле снаружи ограничено сплошной стеной). Таким образом, по строкам и по столбцам можно поддерживать только важные клетки, чтобы быстро искать следующую по направлению. Например, в языке C++ это можно делать при помощи структуры данных `set` и методов `lower_bound`, `upper_bound`. Для текущей точки в зависимости от направления ищем ближайшую стену/ключ/дверь; если ключ — забираем и удаляем; если дверь и можем открыть — открываем

и удаляем; иначе наш шаг закончился. Каждый ключ и каждую дверь мы удалим не более одного раза (при этом не забывайте, что удаление происходит из ряда и из столбца). Получаем решение за $O(H \cdot W \cdot (log(H) + log(W)) + L \cdot log(\max(H, W)))$.

Жюри умеет решать эту задачу за $O((H \cdot W + L) \cdot (\alpha(H) + \alpha(W)))$ с помощью *системы непересекающихся множеств*.

Задача C. Baba is You

В первых двух подзадачах (**20 и 18 баллов**) требуется реализовать то, что просят в условии. Чтобы ответить на запрос, будем переходить по правилам из запроса. Если для какого-то слова есть два правила, выводим «`TOO COMPLEX`». Если мы прошли более 1000 шагов и не попали в слово без правил, выводим «`INFINITE LOOP`».

В третьей подзадаче (**20 баллов**) нужно сделать то же самое, но хранить правила в виде графа в хэш-таблице или любой другой структуре данных. Сложность алгоритма должна быть $O(N \cdot Q \cdot DataStructure)$.

В четвёртой подзадаче (**22 балла**) нужно быстро ходить по графу, в котором из каждой вершины ведёт не более одного ребра. Для этого воспользуемся эвристикой *сжатия путей*. Чтобы ответить на запрос, пройдём по пути до листа, запоминая путь (листом здесь называется вершина, из которой не исходит ребро). После этого перенаправим все рёбра из вершин пути в этот лист. Добавление новых рёбер будем выполнять как обычно.

Это аналогично эвристике сжатия путей, использующейся в *системе непересекающихся множеств* (далее — СНМ), с тем только различием, что мы работает с ориентированными рёбрами. Этот алгоритм работает за $O(N + Q \cdot (logN + logQ))$, доказательство асимптотической сложности здесь не приводится, так как оно совпадает с доказательством для СНМ; его можно найти самостоятельно.

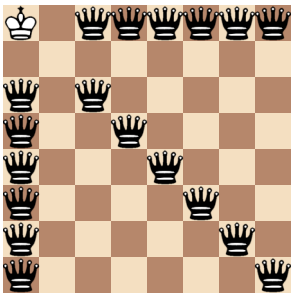
В пятой подзадаче (**20 баллов**) нужно дополнить решение четвёртой подзадачи, научившись обрабатывать ошибки. Чтобы обрабатывать циклы, можно поступать так же, как в СНМ: чтобы провести ребро из листа u в вершину v, запустим сжатие путей из v, и проведём ребро из u в вершину, куда ведёт путь из v. Если образовался цикл, то получится, что мы проведём петлю из u в u. Таким образом, сжатие пути нужно останавливать в листе или в вершине с петлёй, точно так же, как это делается в СНМ.

Чтобы обрабатывать ошибки «`TOO COMPLEX`», будем поддерживать множество *сломанных* вершин — так будем называть вершины, запрос к которым ведёт к этой ошибке. Сломанными являются вершины, из которых исходит два различных ребра, а также все вершины, из который они достижимы. Заметим, что если вершина стала сломанной, она никогда не перестанет быть таковой, так как из неё всегда будет существовать путь к вершине, из которой ведёт два различных ребра. Как только мы «ломаем» вершину, обойдём все еще не сломанные вершины, из которых одна достижимы, и пометим их как сломанные. Для этого будем хранить граф обратных рёбер. Каждую вершину мы пометим как сломанную не более одного раза, поэтому асимптотическая сложность алгоритма останется такой же.

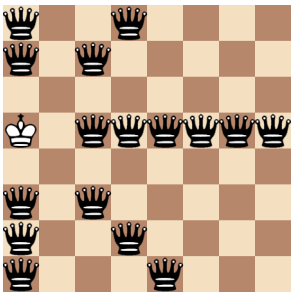
Задача D. Он вам не ферзь

Первую подзадачу (**14 баллов**) труднее решить формулой, чем написать решение второй подзадачи. Однако формула для первой подзадачи используется в последней подзадаче, поэтому приведём её здесь.

1. Если единственный король стоит в углу доски, то ответ равен $M \cdot 3 - 6$.



2. Если единственный король стоит на стороне доски, то ответ равен $M \cdot 3 - 8$.

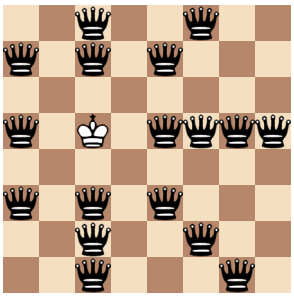


3. Если единственный король стоит в координатах (x, y) не в углу и не в стороне доски (считается, что $(1, 1)$ и (M, M) — координаты противоположных угловых клеток), то ответ равен

$$(M-3) \cdot 2 + min(x-2, y-2) + min(M-x-1, y-2) +$$
$$+min(x-2, M-y-1) + min(M-x-1, M-y-1).$$

Эту формулу можно упростить до

$$M \cdot 3 - 11 + min(x-1, y-1, M-x, M-y) \cdot 2.$$



Чтобы решить вторую подзадачу (**19 баллов**), нужно перебрать все клетки, для каждой клетки перебрать всех королей и проверить, что ферзь из этой клетки бьёт всех королей и сам не находится под боем. Ферзь с координатами (x, y) бьёт короля с координатами (x_i, y_i), если выполняется одно из четырёх условий: $x = x_i$, $y = y_i$, $x + y = x_i + y_i$ или $x - y = x_i - y_i$. Ферзь не находится под боем, если $|x - x_i| \geqslant 2$ или $|y - y_i| \geqslant 2$. Такое решение работает за $O(M^2 \cdot N)$.

Решить третью подзадачу (**20 баллов**) можно несколькими способами, самый простой из них — написать то же самое, что и во второй подзадаче, но обрывать проверку позиции ферзя как только мы находим короля, который не подходит под условия. Оказывается, что такое решение работает за $O(M^2 + M \cdot N)$, так как существует не более $4 \cdot M$ возможных позиций ферзя, которые бьют первого короля; во всех остальных случаях мы выйдем из цикла на первой же итерации.

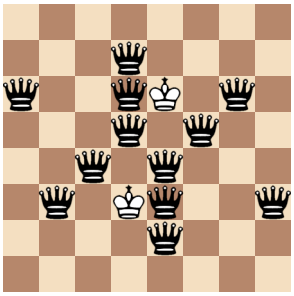
Чтобы решить четвёртую подзадачу (**28 баллов**), нужно перебирать только те позиции, которые бьют первого короля (опять же, их не более $4 \cdot M$), а также быстро проверять выполнение всех

условий. Для этого мы заведём 4 массива счётчиков: первый будет считать количества значений x_i, второй — количества значений y_i, третий — количества значений $x_i + y_i$, четвёртый — количества значений $x_i - y_i$. Также внесём координаты всех королей и всех смежных им клеток в хэш-таблицу.

Чтобы проверить, что ферзь с координатами (x, y) бьёт всех королей, нужно посчитать суммарное количество значений x, y, $x+y$ и $x-y$ в соответствующих массивах. Это число должно равняться n. Кроме того, клетка ферзя должны быть не занята королём и не быть смежной королю. Это можно проверить, сделав один запрос к хэш-таблице. Итого асимптотика решения $O((M+N) \cdot HashTable)$.

Решение пятой подзадачи (**19 баллов**) разбивается на случаи:

1. Есть ровно один король. Формула для этого случая приведена в начале разбора;
2. Есть два короля, не лежащих на одной горизонтали, вертикали и диагонали. Возможно, королей больше, но нам нужно найти двух таких. Существует не более 12 позиций ферзя, бьющих обоих этих королей. Переберём эти позиции и проверим их корректность для всех королей;



3. Все короли лежат на одной прямой (на одной горизонтали, вертикали или диагонали). Если поставить ферзя не на эту прямую, он должен бить первых двух королей. Аналогично предыдущему случаю, нужно проверить не более 12 позиций. Если ферзя поставить на эту прямую, то он автоматически будет бить всех королей; нужно только проверить, что он не стоит рядом с королём. Посчитаем количество клеток на этой прямой и вычтем количество клеток на этой прямой, занятых королём или смежных с королём. Это можно сделать, сложив все такие клетки в хэш-таблицу;
4. Любые два короля лежат на одной прямой, но нет прямой, на которой лежат все короли. Можно рассмотреть любую пару королей, перебрать не более 12 позиций ферзя, не лежащих на общей для них прямой; после чего добавить третьего короля и проверить не более 3 позиций, лежащих на этой прямой и бьющих третьего короля.

Решение 4-го случая можно использовать во 2-м случае, чтобы не искать пару королей, не лежащих на одной горизонтали, вертикали или диагонали. Или же в 4-м случае можно просто рассмотреть все такие расположения королей вручную: есть 8 таких конфигураций для трёх, четырёх или пяти королей.

Итоговая асимптотика решения $O(N \cdot HashTable)$.

Задача E. Сила есть, ум тоже нужен

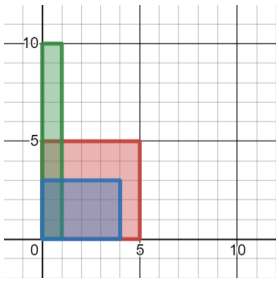
В первой подзадаче (**21 балл**) нужно заметить, что возможных проблем не более 100: проблемы — это пары (x, y), где x и y — натуральные числа от 1 до 10. Нужно перебрать все проблемы, и для каждого человека проверить, какие проблемы он или его подчинённые могут решить. Сделать это можно с помощью поиска в глубину, асимптотическая сложность такого решения $O(N^2 \cdot A \cdot B)$.

Во второй подзадаче (**26 баллов**) нужно заметить, что хоть количество проблем большое, количество значений x не превосходит 100. Значит, перебрать все значения x, и для каждого человека посчитать наибольшее значение y такое, что этот человек может решить проблему (x, y). Если $A_i < x$, то i-й человек не может сам решить проблему с физической сложностью x; если же $A_i \geqslant x$, то этот человек может сам решить проблему (x, B_i). С помощью поиска в глубину посчитаем максимум по всем подходящим значениям B_j, это и будет наибольший y для фиксированного x. Итого решение за $O(N^2 \cdot A)$.

В третьей подзадаче (**23 балла**) нужно отказаться от поиска в глубину. Переберём всех людей циклом от N-го до 1-го. Для каждого человека будем хранить массив из 100 значений — наибольший возможный y для каждого x (по умолчанию 0). Для i-го человека проинициализируем первые A_i значений числом B_i. Затем рассмотрим всех непосредственных подчинённых и обновим текущий массив с помощью массивов подчинённых, заменяя каждое число максимумом. Поскольку суммарное количество непосредственных подчинённых равно $N-1$, решение работает за $O(N \cdot A)$.

Четвёртую подзадачу (**14 баллов**) можно решить с помощью сжатия координат за $O(N^2)$, так как различных значений A не более N. Однако есть другое решение, которое более похоже на решение пятой подзадачи. Заметим, что множество проблем, которое может решить человек или любой из его подчинённых, представляет из себя объединение прямоугольников. Если ввести систему координат и представить i-го человека как прямоугольник с противоположными вершинами $(0,0)$ и (A_i, B_i), то ответ для i-го человека будет равен площади объединения прямоугольников на поддереве.

На рисунке проиллюстрирован пример из условия:



Давайте хранить форму фигуры, задающую ответ i-го человека. Фигура ограничена снизу осью OX и слева осью OY, поэтому достаточно хранить только точки, являющиеся верхними правыми вершинами прямоугольников. Иными словами, будем хранить *выпуклую оболочку* точек (A_i, B_i). Каждый человек инициализируется одной точкой (A_i, B_i). Чтобы посчитать множество точек для человека, нужно объединить его точку и все множества его детей. Это можно сделать по-разному, например, храня точки по возрастанию координаты A_i и строить текущее множество, добавляя туда точки одну за другой. Если в множестве точек лежит точка (A_i, B_i), и мы пытаемся добавить покрывающую её точку (A_j, B_j) с $A_j \geqslant A_i$ и $B_j \geqslant B_i$, удалим точку (A_i, B_i). Если мы пытаемся добавить покрытую точку (A_j, B_j) с $A_j \leqslant A_i$ и $B_j \leqslant B_i$, просто не будем её добавлять.

Можно реализовывать это решение по-разному, но поскольку суммарное количество подчинённых ограничено сверху числом $\frac{N(N-1)}{2}$, решение будет иметь как минимум квадратичную сложность. Можно добиться сложности $O(N^2 logN)$, используя упорядоченное множество, которое есть в стандартной библиотеке некоторых языков программирования.

Чтобы решить пятую подзадачу (**16 баллов**), воспользуемся приёмом «*сливаемые структуры*». Посчитаем размер каждого поддерева. Известно, что количество точек в множестве i-го человека не превосходит размера её поддерева. Чтобы получить ответ для i-го человека, нужно слить множество из одной точки (A_i, B_i) со всеми множествами непосредственных подчинённых i-го человека. Если у i-го человека нет подчинённых, это делается тривиально.

Давайте возьмём за основу множество непосредственного подчинённого с наибольшим размером поддерева. Сделаем его своим множеством, не копируя полностью (например, с помощью функции `move` на языке C++). Если для каждого человека хранится его множество точек, можно скопировать себе множество по указателю или по индексу. После чего переберём все точки во всех остальных множествах, в том числе точку (A_i, B_i). Каждую из них вставим в текущее множество. Здесь важна скорость вставки, вставку нужно выполнять за $O(logN)$. При вставке можно удалить какое угодно количество точек из множества, так как каждая точка удалится не более одного раза.

Получается, что мы за $O(1)$ перемещаем все точки из подчинённого с наибольшим размером поддерева, затем за $O(logN)$ перемещаем каждую оставшуюся точку по одной. Однако, каждый раз когда мы перемещаем точку за $O(logN)$, размер поддерева, где она находится, увеличивается хотя бы вдвое, так как точка перемещается из меньшего поддерева в сумму меньшего с большим. Итого у нас будет не более $O(logN)$ перемещений каждой точки, следовательно, итоговая асимптотика решения $O(Nlog^2 N)$.

Задача F. Трудный выбор

Для первой подзадачи (**10 баллов**) достаточно симулировать процесс и проверять, если для текущей группы по считалочке оказалось, что мы оказались не на стартовой позиции, то это ответ.

Для второй подзадачи (**30 баллов**) нужно свернуть симуляцию в следующее: для текущего b найти первый i такой, что $v \not\equiv 1 \pmod{a_i + 1}$. Запустим перебор, пока не найдём таковой.

Третья и четвёртая подзадачи (**16 баллов каждая**) требуют от нас как-то оптимизировать вычисления: либо в сторону быстрых ответов на запросов, жертвуя асимптотикой по N, либо наоборот - отвечать не быстро, но что-то быстро вычислить по a_i.

Что мы можем сделать в первом случае — давайте вычислим заранее для каждого возможного запроса такую функцию, как mex - первый не встречающийся индекс в наборе, рассмотрев для всех a_i кратные им числа в нашем диапазоне. Получится для каждого числа мы вычислим набор из индексов делителей, а т.к. нам нужен минимальный номер, где стоит не делитель, то mex набора будет ответом.

Если это делать «в лоб», то мы получим вердикт TLE, т.к. для малых a_i очень много кратных. Поэтому разделим числа на две группы — те, по которым mex мы вычисляем, и те, которые мы проверим перебором в запросе. Не трудно подсчитать, что оптимальная граница пройдёт рядом с числом 220, и в таком случае каждое из вычислений у нас займёт по не более чем $2,2 \cdot 10^7$ операций. Ответом будет является $\min(p_1, ..., p_k, mex(b_i - 1))$, где p_j — первая позиция, где встречается j-й не делитель $(b_i - 1)$, меньший 220.

Теперь рассмотрим случай, когда мы хотим отвечать быстро для большого набора чисел, но с малым числом запросов. Давайте сохраним для каждого числа из a_i только первую позицию, где он встретился, а если среди них его не было - число большее n. Ответ на запрос будем вычислять следующим образом - найдём для текущего числа $(b_i - 1)$ все его делители - это точки, которые мы выкалываем из набора, на котором нам нужно найти минимум. Так давайте подсчитаем минимум по отрезкам между этими числами и минимальный из них и будет ответом. Минимум на отрезке можно вычислить за log от длины, найти все делители - за корень от числа, что позволит нам на каждый из запросов отвечать не хуже чем за 2000 операций, а предподсчёт можно сделать не хуже чем за $4 \cdot 10^6$ операций (построение дерева отрезков).

Чтобы решить пятую подзадачу (**15 баллов**), нужно обратить внимание на следующий факт — максимальное число делителей для возможного запроса не превышает 240 (число 720720). Научимся решать за $Q \cdot 241$. Т.к. у нас не более 240 различных делителей, то, перебрав первые 241 различных чисел a_i мы гарантированно завершим перебор. Т.е. оставим в массиве a только различные, оставив только первое вхождение и записав его позицию. Теперь, если мы будем запускать перебор из второй группы он будет работать не за $O(NQ)$, а лишь за $N + Q \cdot 241$.

Для шестой подзадачи (**13 баллов**) давайте будем кешировать наши ответы, и тогда мы ещё сильнее улучшаем время работы до $O(N + Q + X)$, где X — сумма функции количества делителей среди различных b_i. Она в реальности будет ещё быстрее, потому как большинство чисел не будет иметь всех своих делителей на префиксе. Чтобы решение на Python проходило шестую подгруппу нужно оптимизировать I/O при помощи библиотеки *sys*.

Однако есть другое решение для шестой подгруппы — заметим, что если число b_i делится на префикс из чисел a_j, то оно кратно их НОК. При этом если число a_j не увеличивает величину НОК, то оно никогда не будет ответом. А значит мы можем убрать все такие a_j, а также теперь нам достаточно префикса размером $\log b_i$, чтобы дать ответ. В таком случае мы получаем асимптотику $O(N + Q \cdot \log b_i)$, что позволяет решать эту задачу для гораздо больших ограничений a_i и b_i.

Задача G. Ulearn

В первой подзадаче (**17 баллов**) $B < A$, поэтому во время выполнения 2-го этапа Паша не может начать работать над новой задачей; т.е. прежде чем брать следующую задачу, мы для текущей должны выполнить этапы 1-3. Получается, $2A + B$ — это величина времени, за которое делается ровно одна задача, после чего на первый этап выходит следующая, и так далее. Нужно сделать N задач, каждая занимает $2A + B$ времени, и в конце ещё B времени уходит на 4-й этап последней задачи. Итого ответ — $N(2A + B) + B$.

Для решения второй подзадачи (**15 баллов**) нужно отдельно рассмотреть случай, когда $B = A$.

В таком случае мы можем чередовать для пары задач нечётные и чётные этапы (для первой задачи мы делаем 2,3,4 этапы, в то время как у второй выполняются 1,2,3). Если N чётное, то у нас есть $\frac{N}{2}$ пар задач, на каждую пару тратится $5A$ единиц времени. Когда вторая задача выполняет 4-й этап, мы можем для новой пары выполнять 1-й этап первой задачи, а значит мы получаем ответ равный $\frac{N}{2} \cdot 4A + A = (2N + 1)A$. Если N нечётное, то нужно в конце добавить еще $3A$ времени на последнюю задачу. Получается ответ равный $\frac{N+1}{2} \cdot 4A = (2N + 2)A$. Нетрудно заметить, что и чётный и нечётный случай объединяются в одну формулу $(2N + 1 + (N \ mod \ 2))A$.

Чтобы решить третью подзадачу (**16 баллов**), рассмотрим отдельно 2 случая: $B = 2A$ и $A < B < 2A$.

1. Пусть $B = 2A$. Теперь мы можем параллельно работать уже с тремя задачами. Время выполнения одной такой группы составит $8A$ единиц времени, причём следующая группа задач может создаваться уже в течение последних $2A$ единиц. Получаем, что, если N кратно трём, то ответ $\frac{N}{3} \cdot 6A + 2A = (2N + 2)A$. Если же нет, то у нас есть неполная группа в размере 1 или 2 задач. Для них ответ будет равен $\lfloor \frac{N}{3} \rfloor \cdot 6A + XA$, где X равен 6, если размер неполной равен одному, и 7, если двум.

2. Пусть $A < B < 2A$. Этот случай аналогичен случаю $B = A$ в том смысле, что Паша может работать сразу над двумя задачами. Одна пара выполняется $3A + 2B$ времени, и в последние B единиц времени мы можем начать делать следующую пару. Получается формула $\frac{N}{2} \cdot (3A + B) + B$ для чётного количества и $\frac{N+1}{2} \cdot (3A + B) - A + B$ для нечётного.

Для решения четвёртой подзадачи (**26 баллов**) нужно жадно симулировать решение задач. Каждую следующую задачу нужно начать как можно раньше. Для этого нужно поддерживать последний и предпоследний моменты времени, когда Паша освободился. Мы всегда можем начать делать задачу после последнего момента, а после предпоследнего — не всегда. Дело в том, что последний промежуток времени, когда работает Паша — это 3-й этап нескольких задач; а освобождается в предпоследний раз он после того, как закончит 1-й этап. Нужно поддерживать также начало этого промежутка, чтобы понять, сможет ли Паша уместить ещё одну задачу. В отличие от других подзадач, сложность работы этого решения составляет $O(N)$.

Для решения пятой подзадачи (**26 баллов**) нужно все случаи свести к одному. У нас всегда есть группы задач, которые мы делаем параллельно, у них 1-е этапы идут один за другим. Паша может параллельно работать над $\lfloor \frac{B}{A} \rfloor + 1$ задачами. Если в группе между 1-м этапом последней задачи и 3-м этапом первой задачи есть «зазор», то он меньше A и его ничем не заполнить.

Составим формулу. Для удобства введём обозначение $C = \lfloor \frac{B}{A} \rfloor + 1$. Одна группа задач находится на этапах 1-3 в течение $2CA + (B \ mod \ A)$ времени. Всего есть $\lfloor \frac{N}{C} \rfloor$ групп из C задач и, возможно, одна неполная группа. В неполную группу входит $N \ mod \ C$ задач, обозначим это число как D. Время выполнения неполной группы на этапах 1-3 равно $(C + D)A + (B \ mod \ A)$.

Остаётся прибавить B — время выполнения 4-го этапа последней задачи. Получаем формулу

$$\frac{N}{C} \cdot (2CA + (B \ mod \ A)) + B,$$

если N делится на C, иначе

$$\left\lfloor \frac{N}{C} \right\rfloor \cdot (2CA + (B \ mod \ A)) + (C + D)A + (B \ mod \ A) + B.$$

Всё это можно упростить до одной формулы

$$\left\lceil \frac{N}{C} \right\rceil \cdot (CA + (B \ mod \ A)) + NA + B.$$

Вузовско-академическая олимпиада
по программированию на Урале

2020-2021 учебный год

Критерии определения призеров и победителей

Вузовско-академическая олимпиада по программированию на Урале

Критерии определения призеров и победителей заключительного этапа 2020-2021 учебного года

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

Степень диплома	**Разбалловка**	**Участников**
		267
5-11 классы	Максимум 700 баллов	
I	От 520 баллов и выше	21
II	От 410 до 519 баллов включительно	22
III	От 354 до 409 баллов включительно	23
Количество призеров и победителей заключительного этапа		66
Процент призеров и победителей от общего числа участников заключительного этапа		25%