Задача A. Вписанная окружность

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В этой задаче требуется всего лишь найти координаты центра вписанной окружности заданного треугольника.

Формат входных данных

Во вводе содержится описание одного или нескольких треугольников. Первое число N обозначает количество треугольников, для которых необходимо решить задачу. Далее следуют N строк с координатами вершин треугольников (шесть чисел x_1 y_1 x_2 y_2 x_3 y_3). Все треугольники во входном файле невырожденные, а координаты целые и по модулю не превосходят 1000.

Формат выходных данных

Для каждого треугольника из ввода необходимо выдать в отдельной строке два числа, разделённых пробелом, — x- и y-координаты центра вписанной окружности. Координаты необходимо выводить не менее чем с четырьмя точными знаками после десятичной точки.

Пример

стандартный ввод	*стандартный вывод*
1 0 1 1 0 0 0	0.292893 0.292893

Задача B. Ханойские башни 2

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ханойские башни — популярная головоломка. Она состоит из трёх стержней и n дисков различного диаметра. В центре каждого диска находится отверстие для того, чтобы нанизывать диск на стержень. Изначально все диски нанизаны на первый стержень, причём сверху расположен самый маленький диск, под ним диск побольше и так далее; снизу лежит самый большой диск. За одно перекладывание разрешается снять один диск сверху любого стержня и нанизать его сверху на другой стержень. При этом нельзя нанизывать диск на стержень, на котором верхний диск имеет меньший диаметр; на пустой стержень можно нанизывать любой диск. Цель — перекладывать диски таким образом, чтобы перенести их все на третий стержень.

Напишите программу, которая решает головоломку в общем виде: по заданному количеству дисков и данному начальному положению находит последовательность перекладываний, позволяющую перевести их в данное конечное положение. Количество перекладываний должно быть минимально возможным.

Формат входных данных

В первой строке записано целое число n — количество дисков ($1 \leqslant n \leqslant 15$). Во второй строке записано n целых чисел через пробел — номера стержней, на которых изначально лежат диски. В третьей строке также записано n целых чисел через пробел — номера стержней, на которых должны оказаться диски. В этих строках диски перечислены от маленьких к больши́м. Если несколько дисков в начальном или конечном положении лежат на одном стержне, это означает, что внизу лежит самый большой из них, на нём — самый большой из оставшихся, и так далее; наверху лежит самый маленький диск.

Формат выходных данных

В первой строке выведите m — минимальное число перекладываний. В следующих m строках выведите описания операций перекладывания по одному на строке. Описание перекладывания должно состоять из двух целых чисел, разделённых пробелом — номеров стержня, с которого снимается диск, и стержня, на который он нанизывается. Если существует несколько способов решить головоломку за минимальное число перекладываний, можно выводить любой из них. Если решения не существует, выведите в первой строке число -1.

Примеры

стандартный ввод	*стандартный вывод*
2 1 1 3 3	3 1 2 1 3 2 3
3 1 2 3 2 3 1	5 1 2 3 1 2 1 2 3 1 2

Задача C. Число из спичек

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вы умеете выкладывать на столе некоторые цифры от 0 до k с помощью какого-то количества спичек. Ваша задача — построить максимально большое число из n спичек. Например, если вам нужно 6 спичек для цифры 0, 7 спичек для цифры 1 и 8 спичек для цифры 2, то максимальное число, которое можно сложить из 21 спички — это 210.

Формат входных данных

В первой строке даны числа n и k ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant 9$). Во второй строке задано $k + 1$ число от 1 до 50 — количество спичек, требуемое на цифры от 0 до k. Гарантируется, что из n спичек можно выложить хотя бы одну из цифр.

Формат выходных данных

Выведите максимальное число, которое можно построить. Лидирующие нули в выводе недопустимы. Можно использовать не все n спичек, а только часть.

Пример

стандартный ввод
21 2 6 7 8
стандартный вывод
210

стандартный ввод
30 2 5 23 24
стандартный вывод
20

стандартный ввод
1 3 1 5 3 2
стандартный вывод
0

стандартный ввод
50 9 1 1 1 1 1 1 1 1 1 1
стандартный вывод
99999999999999999999999999999999999999999999999999

Задача D. Программирование с кружкой

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася — первоклассный программист, он свободно владеет десятипальцевым набором, что позволяет ему набирать любой текст с умопомрачительной скоростью. Кроме программирования, Вася очень любит пить чай. И вот только он налил себе полную кружку восхитительного чёрного чая и сел читать цикл постов про портирование Windows 3.1 на чайную ложку, как неожиданно ему на почту пришло уведомление — уже через 0.35 миллисекунд начинается очередной раунд на Topcoder! До начала раунда Вася успел только ввести логин и пароль и открыть условия.

Поскольку Вася не просто программист, но ещё и бывалый ACM-щик, на прочтение условий и придумывание решений ему требуется пренебрежимо мало времени. Остаётся только вбить код, Вася и это сделал бы моментально, но у него налит чай! Разумеется, просто оставить его остывать он не может, поэтому одной рукой он всегда должен держать кружку. Но поскольку одна рука Васи теперь занята, он не может использовать всю мощь десятипальцевого набора. Свободной от кружки рукой Вася по-прежнему может набирать символы, но не все с одинаковой скоростью. Как известно, при десятипальцевом наборе любая клавиша на клавиатуре относится к левой или правой руке (для простоты опустим подробности с заглавными буквами и пробелом — писать без пробелов в любом случае быстрее). Поэтому, если Вася держит кружку в своей левой руке, то код он набирает правой рукой, и символы с клавиш, относящихся к правой руке, он набирает со своей обычной скоростью в один символ за A миллисекунд. Однако нажимать правой рукой клавиши с части клавиатуры, соответствующей левой руке, он так же быстро не может, и в этом случае он должен тратить на набор целых B миллисекунд.

Весь код, который Васе необходимо напечатать, можно представить в виде последовательности символов, причём про каждый символ известно, к какой части клавиатуры он относится. Мастерство Васи позволяет избегать исправлений и рефакторинга, поэтому ему нужно просто по очереди ввести все символы последовательности. Кроме того, для оптимизации процесса Вася может перемещать кружку из одной руки в другую, на это у него уходит C миллисекунд, и печатать он ничего в это время не может. Вася уже посчитал, какое минимальное время ему потребуется, чтобы написать весь код, а сможете ли вы?

Изначально Вася держит кружку в левой руке. После ввода текста кружка может оказаться в любой руке.

Формат входных данных

В первой строке заданы три целых числа A, B и C — время в миллисекундах на набор символа, относящегося к свободной руке, на набор символа, относящегося к несвободной руке, и на перемещение кружки из одной руки в другую. Каждое из этих чисел положительно и не превосходит 1000, кроме того, $A \leqslant B$.

Во второй строке — количество символов, которые необходимо напечатать Васе, их не меньше одного и не больше $2 \cdot 10^5$.

В третьей строке задана сама последовательность — строка из символов «`l`» и «`r`»: «`l`» соответствует клавише, относящейся к левой руке, «`r`» — к правой.

Формат выходных данных

Выведите одно число — минимальное время в миллисекундах, за которое Вася успеет ввести весь текст.

Примеры

стандартный ввод	*стандартный вывод*
1 2 2 8 rrlrrlll	11
1 2 5 5 rlrlr	7

Задача E. Не делитель

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дано целое положительное число n. Найдите любое целое число от 1 до n включительно, не являющееся делителем n, или сообщите, что таких чисел нет.

Формат входных данных

В единственной строке входных данных находится целое положительное число n ($1 \leqslant n \leqslant 9$).

Формат выходных данных

Если среди чисел от 1 до n включительно нет чисел, не являющихся делителями n, то выведите `-1`. Иначе выведите целое число от 1 до n включительно, не являющееся делителем n. Если подходит несколько ответов, то можно вывести любой.

Примеры

стандартный ввод	стандартный вывод
1	-1
2	-1
6	4
9	7

Пояснения к примерам

В первом примере из условия ответ `-1`, так как 1 делится на 1.

Во втором примере из условия ответ `-1`, так как 2 делится на 1 и 2.

В третьем примере из условия подходят ответы `4` и `5`.

Наконец, в последнем примере из условия подходят ответы `2`, `4`, `5`, `6`, `7` и `8`.

Задача F. Произведение многочленов

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Заданы два многочлена от переменной `x`. Найдите их произведение.

Формат входных данных

В первой строке записан многочлен P, а во второй — многочлен Q. Степень каждого из них строго меньше $65\,536$. Коэффициенты многочленов целые и не превосходят $100\,000$ по абсолютной величине.

Далее описан формат записи многочлена.

Одночлены, из которых состоит многочлен, отделены друг от друга знаком арифметической операции «+» или «−», вокруг которого стоят одиночные пробелы. Одночлены перечислены в порядке от старших к младшим. Многочлен не может содержать двух одночленов, соответствующих одной и той же степени `x`. Степени `x` с нулевым коэффициентом опускаются.

Одночлен записывается в общем случае в виде «c`x^`p». Но, если $p = 1$, запись имеет вид «c`x`», а если $p = 0$, запись имеет вид «c». Если $|c| = 1$, а $p \neq 0$, символ «1» опускается. Коэффициент c записывается как строго положительный, кроме, возможно, коэффициента у самого старшего одночлена: у всех остальных знак учитывается в арифметической операции перед ними.

Если многочлен тождественно равен нулю, он записывается как «0».

Для лучшего понимания формата посмотрите на примеры.

Формат выходных данных

Выведите многочлен $P \cdot Q$ в том же формате, что и заданные многочлены. Соблюдайте формат как можно более точно! Проверка производится посимвольно.

Примеры

стандартный ввод	*стандартный вывод*
`x^2 - 2x + 1` `-5x^3 - x`	`-5x^5 + 10x^4 - 6x^3 + 2x^2 - x`
`9999` `9999`	`99980001`

Задача G. Ординальные числа

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ординальное число — это обобщение понятия натурального числа (здесь и далее в этой задаче 0 считается натуральным числом). Ординальные числа, соответствующие натуральным числам, можно определить рекурсивно:

> Ординальное число, соответствующее натуральному числу n — это множество, элементами которого являются ординальные числа, соответствующие натуральным числам 0, 1, 2, …, $n-1$.

В частности, пустое множество $\varnothing = \{\}$ — ординальное число, соответствующее натуральному числу 0, единице соответствует множество $\{\varnothing\} = \{\{\}\}$, двойке — множество $\{\varnothing, \{\varnothing\}\} = \{\{\}, \{\{\}\}\}$ и так далее. Удобно записывать ординальное число просто как соответствующее ему натуральное число, например, $4 = \{0, 1, 2, 3\}$. Вообще, $n = \{0, 1, \ldots, (n-1)\}$.

Такое определение удобно тем, что его можно расширить на бесконечные множества. К примеру, наименьшее бесконечное ординальное число ω определяется как множество, содержащее все конечные ординальные числа. Следующее за ним число (будем записывать его как $\omega + 1$) равно $\{\omega, 0, 1, \ldots\}$ или, другими словами, $\omega + 1 = \omega \cup \{\omega\}$, и так далее. В общем случае по любому ординальному числу α можно определить следующее ординальное число $\alpha + 1 = \alpha \cup \{\alpha\}$. Однако, предыдущее ординальное число определено не для всех чисел, например, для ω не существует такого γ, что $\omega = \gamma \cup \{\gamma\}$.

Игорь выписал на доске некоторое конечное ординальное число. Запись была корректной и состояла из открывающих и закрывающих фигурных скобок, а также запятых. Формат записи рекурсивно определяется так: запись каждого множества начинается с открывающей фигурной скобки, далее по одному разу следуют записи всех элементов этого множества в произвольном порядке, разделённые запятыми, а в конце стоит закрывающая фигурная скобка. К примеру, ординальное число 0 записывается как «`{}`», 1 — как «`{{}}`», 2 — как «`{{},{{}}}`» или «`{{{}},{}}`» и так далее.

Улучив момент, Владислав подкрался к доске и стёр один символ. Запись больше не является корректной! Это несколько расстраивает Игоря. Установив виновника, Владимир Михайлович попросил Владислава срочно исправить положение. Помогите Владиславу восстановить исходную запись.

Формат входных данных

В первой строке ввода задана запись ординального числа, в которой отсутствует ровно один символ. Длина этой записи — от 1 до $3 \cdot 10^6$ символов.

Формат выходных данных

Выведите одну строку — правильную запись ординального числа, полученную добавлением одного символа в любое место заданной записи. Если правильных ответов несколько, выведите любой из них.

Примеры

стандартный ввод	*стандартный вывод*
`{{{}}{}}`	`{{{}},{}}`
`{{{}},}}`	`{{{}},{}}`
`{{{},{}}`	`{{{}},{}}`

Пояснения к примерам

Во всех трёх тестах после добавления одного символа получается ординальное число 2.

В первом тесте нужно добавить запятую, во втором — открывающую фигурную скобку, в третьем — закрывающую фигурную скобку.

Задача H. И снова сумма...

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

Реализуйте структуру данных, которая поддерживает множество S целых чисел, к которому поступают следующие запросы:

- `add (`i`)` — добавить в множество S число i (если оно там уже есть, то множество не меняется);
- `sum (`l`, `r`)` — вывести сумму всех элементов x из S, которые удовлетворяют неравенству $l \leqslant x \leqslant r$.

Формат входных данных

Исходно множество S пусто. Первая строка входных данных содержит целое число n — количество операций ($1 \leqslant n \leqslant 300\,000$). Следующие n строк содержат операции. Каждая операция выглядит либо как «`+` i», либо как «`?` l r».

Операция «`?` l r» задаёт запрос «`sum (`l`, `r`)`».

Если операция «`+` i» находится в начале входных данных или следует непосредственно после другой операции «`+`», то она задаёт запрос «`add (`i`)`». Если же она следует непосредственно после запроса «`?`», и результат этого запроса был y, то выполняется операция «`add (`v`)`», где $v = (i + y) \bmod 10^9$.

Во всех операциях параметры лежат в интервале от 0 до 10^9 включительно.

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Пример

стандартный ввод	*стандартный вывод*
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Задача I. Различные подстроки 3

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Подстрокой строки $\mathcal{S} = s_1 s_2 \dots s_n$ называется непрерывная подпоследовательность символов этой строки $s_i s_{i+1} s_{i+2} \dots s_{j-1} s_j$.

Дана строка. Сколько различных подстрок, не считая пустой, она содержит? Выведите все эти подстроки по одному разу в лексикографическом порядке. Рядом с каждой подстрокой выведите, сколько раз она встречается в строке $\mathcal{S}$.

Формат входных данных

В первой строке ввода задана строка длины от 1 до 100 символов, включительно. Строка состоит из маленьких букв английского алфавита.

Формат выходных данных

В первой строке выведите одно число r — количество различных подстрок данной строки, не считая пустой. В следующих r строках выведите сами эти подстроки в лексикографическом порядке. После каждой подстроки через пробел должно следовать целое число — сколько раз эта подстрока встречается в строке $\mathcal{S}$.

Примеры

стандартный ввод	*стандартный вывод*
aab	5 a 2 aa 1 aab 1 ab 1 b 1
da	3 a 1 d 1 da 1

Задача J. Доска Софи

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мебибайт

У Софи есть бесконечная доска, разделённая на квадратные клетки. Каждая клетка имеет свой цвет: n клеток покрашены в чёрный цвет, остальные — в белый. На доске введена система координат: каждая клетка доски имеет целые координаты, а x-координаты соседних клеток в любом ряду и y-координаты соседних клеток в любом столбце отличаются на единицу.

Множество клеток A называется связным по стороне, если для любой пары клеток из A существует путь от одной из них до другой, проходящий только по клеткам из A, и в этом пути любые две последовательные клетки имеют общую сторону.

Софи хочет узнать, является ли множество белых клеток доски связным по стороне. Помогите ей это сделать.

Формат входных данных

В первой строке записано целое число n — количество чёрных клеток ($1 \leqslant n \leqslant 100\,000$). В следующих n строках записаны координаты чёрных клеток. В i-й из них записаны через пробел два целых числа x_i и y_i — координаты i-й чёрной клетки ($-10^9 \leqslant x_i, y_i \leqslant 10^9$). Все заданные чёрные клетки различны.

Формат выходных данных

Выведите «`yes`» (без кавычек), если множество белых клеток является связным по стороне, и «`no`» иначе.

Примеры

стандартный ввод	*стандартный вывод*
3 0 0 1 0 0 1	yes
8 0 0 0 1 0 2 1 2 2 2 2 1 2 0 1 0	no
2 0 0 2 0	yes