Для всех задач:

Имя входного файла:	***input.txt***
Имя выходного файла:	***output.txt***
Ограничение по памяти:	***256 Мб***

Задача 1. Сладкие воспоминания

Ограничение по времени на 1 тест: ***2 сек.***

Суетливо слюнявя пальцы, Паниковский разложил все деньги, изъятые у Корейко, на несколько кучек. Потом великий комбинатор изъял все эти деньги у него, и тут-то сын лейтенанта Шмидта бесповоротно загрустил. Окончательно его подкосило фиаско с золотыми гирями. Все, что оставалось, это предаваться сладким воспоминаниям, в какой кучке сколько было денег. Чтобы это не забыть, великий слепой выписал эти числа в столбец одно за другим. В следующий столбик он выписал разности между двумя соседними числами, в третий — разности между числами из второго столбца и так далее, пока не получил в последнем столбце одно-единственное число. Таким образом, все вычисления он производил по формуле

$y[k+1, i] = y[k, i + 1] – y[k, i]$,

где $y[k, i]$ — элемент таблицы, k — номер столбца, i — номер строки.

Вам необходимо получить это же самое число, вычисленное по простому модулю p.

Входные данные

В первой строке входного файла записано два целых числа N и p, где N — количество кучек денег, а p — простое число ($1 \le N \le 10^5$, $2 \le p < 2 \cdot 10^9$). Во второй строке записано N неотрицательных целых чисел, не превосходящих 10^9 – количество денег в каждой кучке по порядку.

Выходные данные

В выходной файл необходимо вывести последний (крайний правый) элемент полученной таблицы, вычисленный по простому модулю p.

Пример

input.txt	*output.txt*
5 7 2 6 7 3 10	4

Задача 2. Пьяный матрос (Division 1 only)

Ограничение по времени на 1 тест: ***3 сек.***
для задач на Java - ***8 сек.***

Однажды пьяный матрос заблудился в лесу. Лес представляет собой прямоугольник размера $N \times M$ метров со сторонами, параллельными осям координат. Полагаем, что левый нижний угол этого прямоугольника совмещен с началом координат. Тогда деревья будут расположены во всех точках с целочисленными координатами, лежащих внутри или на границе прямоугольника.

Матрос пытается выбраться из леса. Но поскольку он пьян, и ориентироваться в лесу не может, он просто переходит от дерева к дереву, каждый раз выбирая следующее дерево равновероятно случайным образом среди четырех ближайших соседей дерева, около которого он стоит. При этом если он доходит до дерева, растущего на границе леса, он соображает, что нашел выход из леса и благополучно из него выбирается.

Однако помешать благополучному возвращению матроса из леса может медведь. Медведь сидит на дереве с координатами (x_b, y_b). Если матрос оказывается под этим деревом… В общем, его путь также заканчивается.

Ваша задача — по заданным размерам леса N и M, по координатам медведя (x_b, y_b) и начальной позиции матроса (x_s, y_s) посчитать математическое ожидание количества переходов матроса от дерева к дереву до того или иного окончания его пути.

Напомним, что математическое ожидание $E(\xi)$ дискретной случайной величины ξ равно конечной либо бесконечной сумме

$$E(\xi) = \sum_{\xi} P(\xi) \cdot \xi ,$$

где $P(\xi)$ – вероятность случайной величины принять значение ξ. Можно показать, что в данном случае, когда ξ равна количеству переходов матроса от дерева к дереву до окончания пути, ряд в выражении выше сходится, а значит математическое ожидание $E(\xi)$ определено корректно.

Входные данные

В первой строке входного файла записаны два целых числа N и M ($1 \le N, M \le 700$). Во второй строке записаны два целых числа x_b и y_b — координаты медведя ($0 \le x_b \le N$, $0 \le y_b \le M$). В третьей строке содержится пара целых чисел x_s, y_s — начальная позиция матроса в лесу ($0 \le x_s \le N$, $0 \le y_s \le M$).

Выходные данные

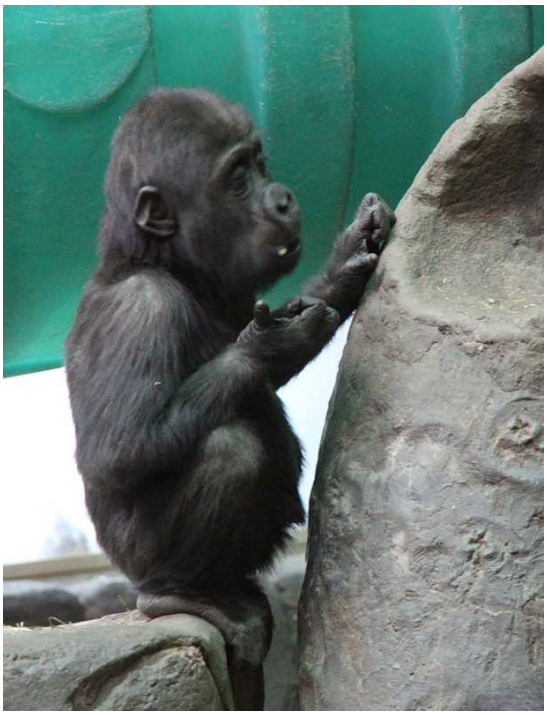
В выходной файл необходимо вывести единственное число — математическое ожидание количества переходов матроса между деревьями $E(\xi)$, считая, что он начинает свой путь в точке (x_s, y_s) и заканчивает либо на краю леса, либо под деревом с медведем. Ответы необходимо выдавать с абсолютной либо относительной точностью не хуже 10^{-8}.

Пример

input.txt	*output.txt*
2 4 **0 0** **1 1**	**1.4285714286**

Задача 3. Штукатурка

Ограничение по времени на 1 тест: ***2 сек.***



Вася наказан. Его поставили лицом к стене подумать о своем поведении. От нечего делать, стоя у стены, Вася решил поковырять пальцем штукатурку. Он хочет получить круг максимального радиуса, но шевелить он может только пальцем, рука и кисть должны оставаться неподвижными. Если родители заметят это его занятие, то еще больше накажут.

Вытянутым пальцем он намечает границу круга максимального радиуса, а затем отковыривает штукатурку из его середины. Основание пальца зафиксировано в точке, которая находится на расстоянии S см от стены. Длина пальца L см. Вася может отклонять палец от перпендикуляра к стене в горизонтальной плоскости на угол α, а в вертикальной — на угол β. В соответствии с этими правилами Вася может отклонять палец не дальше поверхности эллиптического конуса, вершина которого совпадает с основанием его пальца, а ось конуса перпендикулярна стене.

Входные данные

В первой строке входного файла записаны четыре целых числа S, L, α и β ($0 < S, L \le 15$, $0 \le \alpha, \beta \le 80$). Углы задаются в градусах.

Выходные данные

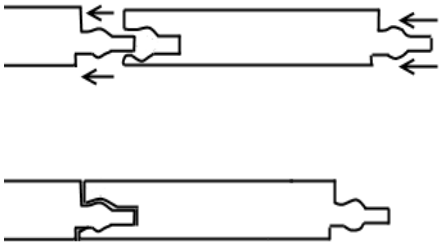
В выходной файл необходимо вывести площадь круга максимального радиуса, который Вася сможет отковырять, с точностью до 10^{-4}. Если палец слишком длинный, то он может проткнуть стену довольно глубоко, но Васю интересует только площадь круга на поверхности стены.

Пример

input.txt	*output.txt*
10 12 30 30	**104.7197**

Задача 4. Ламинат

Ограничение по времени на 1 тест: ***2 сек.***



Ламинат представляет собой прямоугольные пластины с замками на краях, при помощи которых они стыкуются вместе. Замки устроены следующим образом: на одной стороне панели имеется так называемый «шип», на другой соответственно паз. В процессе сборки шип забивается в паз.

Требуется уложить ламинатом пол в заданной комнате. Комната имеет прямоугольную форму, при этом она расположена так, что стены параллельны сторонам света. На стене с восточной стороны комнаты есть окно. В одном из углов комнаты будет установлен прямоугольный подиум, на который не надо укладывать ламинат. Известно, что одна сторона подиума примыкает к стороне с окном, а другая к соседней стене.

Следует использовать прямую укладку ламината длинной стороной вдоль направления запад-восток.

Мы — строители-дилетанты, и в нашем наборе инструментов не оказалось линейки. Поэтому за единицу измерения приняли размер меньшей стороны плитки ламината. К счастью, все размеры оказались кратные.

Если плитки друг с другом необходимо соединять замками, то плитку со стеной можно стыковать и без замков. Поэтому, если в конце ряда осталось места меньше, чем размер плитки, то её можно разрезать и положить стороной без замка к стене или к подиуму. Так же можно начинать ряд с разрезанной плитки, положенной стороной без замка к стене (или к подиуму). Стоит учитывать, что после разрезания одной плитки полученные из неё куски можно использовать не более чем в двух рядах. Один кусок положить у восточной стены или у подиума, а другой — у западной. Разрезать плитку можно несколько раз. Резать можно только поперек длинной стороны. И не надо забывать, что линейки у нас нет, поэтому разрезанные куски имеют размер, кратный нашим «попугаям». Из эстетических соображений отрезанные куски, которые получились квадратными, использовать нельзя.

Входные данные

В первой строке входного файла записаны два целых числа N и M — размеры комнаты в попугаях, где N размер стены с окном ($2 \le M \le 10^3$, $2 \le N \le 500$).

Во второй строке заданы размеры подиума S и K, где S — размер стороны подиума, которая примыкает к стене с окном ($1 \le S < N$, $1 \le K \le M - 2$).

В следующей строке задан размер плитки ламината L ($3 \le L \le 10$).

Выходные данные

В выходной файл необходимо вывести минимальное число плиток, необходимых для укладки всей комнаты.

Примеры

input.txt	*output.txt*
10 10 **5 5** **5**	**15**
4 8 **2 1** **6**	**5**

Задача 5. ЙААЗЬ

Ограничение по времени на 1 тест: ***2 сек.***

На охранном предприятии «Рыба моей мечты» для патрулирования территорий вводится новая должностная инструкция под кодовым названием «ЙААЗЬ». Для оптимизации процесса любая прямоугольная территория разбивается так, чтобы представлять собой таблицу размером $2^N \times 2^N$.

Примеры разбиения:

$N = 0$ $N = 1$ $N = 2$

Таким образом, каждой ячейке на уровне $(N - 1)$ соответствует четыре «дочерних» ячейки на уровне N. Разбиение идёт до максимального уровня M.

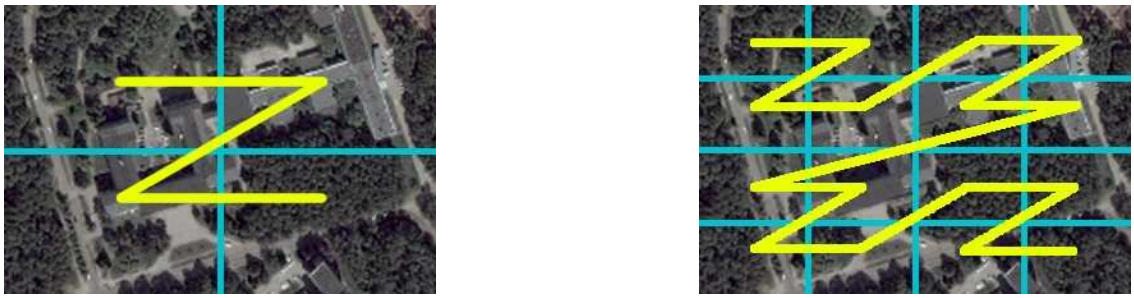
В новой инструкции также фиксируется порядок патрулирования:

1. Охранник начинает обход с уровня $N = 0$.
2. Если охранник находится в ячейке на уровне $N < M$, то он переходит в одну из четырех соответствующих ей «дочерних» ячеек на уровне $(N + 1)$ в порядке: «левая верхняя», «правая верхняя», «левая нижняя», «правая нижняя». При этом охранник в итоге переходит в каждую ячейку ровно один раз.
3. После просмотра ячейки на уровне N $(N > 0)$ охранник сразу возвращается в «родительскую» ячейку на уровне $(N - 1)$.
4. Охранник, оказавшись в ячейке на уровне $N = M$, должен сообщить диспетчеру порядковый номер ячейки (номера считаются только для ячеек на уровне M).

При этом охранник в действительности ходит только по ячейкам максимального уровня, ячейки других уровней он посещает мысленно.

Примеры обхода территории для максимального уровня M в порядке номеров, которые охранник сообщает диспетчеру:

$M = 1$ $M = 2$



Диспетчеру для каждого номера, который ему сообщает охранник, необходимо знать координаты (*x*, *y*) ячейки в таблице. Ось *X* направлена вниз, ось *Y* – вправо, левая верхняя ячейка имеет координаты (0, 0).

Входные данные

В первой строке входного файла записано единственное целое число S – количество охранников ($1 \le S \le 10^5$) .

В следующих S строках записано по два целых числа: M_i и k_i, ($0 \le M_i \le 15$, $0 \le k_i < 2^{2M_i}$). M_i определяет максимальный уровень разбиения территории, а k_i — порядковый номер ячейки на максимальном уровне, в которой находится i-й охранник.

Выходные данные

В выходной файл для каждого охранника в том же порядке, что и во входном файле, нужно выдать по два целых числа на строке — координаты (*x*, *y*) ячейки таблицы на максимальном уровне, в которой находится этот охранник.

Пример

input.txt	*output.txt*
5 **0 0** **2 0** **2 15** **2 7** **2 8**	**0 0** **0 0** **3 3** **1 3** **2 0**

Задача 6. На дрезине

Ограничение по времени на 1 тест: ***2 сек.***
для задач на Java - ***3 сек.***

Геральд Геральдович очень любит путешествия. Но больше всего он любит железнодорожный транспорт. К сожалению, на свой поезд он успел опоздать, поэтому теперь между станциями ему придётся двигаться на дрезине. Всё бы хорошо — работать он любит, физический труд ему не чужд, но дьявол, как всегда, кроется в мелочах.

В данный момент Геральд Геральдович ждёт очень важный звонок на свой сотовый телефон. А его оператор «Еле-Еле» обладает очень сильной чувствительностью к выходам за зоны действия вышек. Поэтому возникает такая проблема — если уровень сигнала низок, Геральду Геральдовичу приходится поднимать левой рукой свой телефон как можно выше, и приводить дрезину в движение одной правой, что заметно замедляет его действия.

К счастью, он раздобыл не только карту железных дорог, но и карту расположения вышек сотовой связи. Для каждой вышки он знает её координаты и радиус действия. Теперь, со всеми этими данными, ему осталось только проложить маршрут, чтобы ехать оптимально. А поскольку навигаторы ещё не умеют работать с железнодорожными направлениями, придётся потрудиться вам.

Карта железных дорог задана списком координат станций на плоскости и набором пар станций, между которыми проложен железнодорожный путь. Путь следует считать отрезком, соединяющим две станции. Если два пути пересекаются в геометрическом смысле вне станции, то следует считать, что переход на другой путь невозможен. Аналогичное правило действует и для станций — если путь между станциями A и B проходит через станцию C в геометрическом смысле, но пути AC и CB не значатся на карте, считается, что переход с пути AB на станцию C невозможен. Путь является двусторонним — если в карте значится путь AB, то по нему можно попасть как из A в B, так и из B в A.

Карта вышек задана списком координат вышек с радиусом действия. Считается, что связь в произвольной точке есть, если она попадает в радиус действия хоть одной вышки, иначе связи нет. Если отрезок пути касается окружности, покрытой вышкой, то считается, что в точке касания связи нет.

Ваша цель — найти такой маршрут между заданными станциями, что суммарное расстояние, на котором Геральд Геральдович будет находиться вне зоны действия сети, будет минимальным. Чтобы поездка не была однообразной, нельзя посещать одну и ту же станцию более одного раза. Опять же, посещением станции C считается проезд по пути AC, а не проезд по пути AB, проходящему через станцию C в геометрическом смысле.

Входные данные

В первой строке входного файла записаны три целых числа N, M и K — количество станций, путей и вышек, соответственно ($2 \le N \le 100$, $1 \le M \le N \cdot (N–1) / 2$, $1 \le K \le 100$). Все станции пронумерованы числами от 1 до N.

Во второй строке указаны номера станций начала и конца маршрута — целые числа S и T ($1 \le S, T \le N$).

Следующие K строк содержат описания вышек. Каждая вышка задается тремя действительными числами X_k, Y_k, R_k , записанными на одной строке через пробел. Первые два числа —

это координаты k-ой вышки, а третье — радиус её действия ($-1000 \le X_k, Y_k \le 1000$, $0 < R_k \le 1000$, $1 \le k \le K$).

В следующих N строках записаны координаты станций в порядке их нумерации. Координаты задаются д вумя действительными числами X_i и Y_i, записанными на одной строке через пробел ($-1000 \le X_i, Y_i \le 1000$, $1 \le i \le N$). Станции отстоят друг от друга не менее чем на 10^{-6}.

В следующих M строках описываются пути. Каждый путь задается парой целых чисел A_j и B_j, записанных на одной строке через пробел,— номерами станций, между которыми он проходит ($1 \le A_j < B_j \le N$, $1 \le j \le M$). Гарантируется, что между двумя станциями не может быть более одного пути.

Выходные данные

В первую строку выходного файла необходимо вывести одно вещественное число L — минимальное суммарное расстояние, которое придется преодолеть вне зоны действия сети. Ответ считается верным, если его абсолютная или относительная погрешность не превосходит 10^{-3}.

Во второй строке должны быть записаны через пробел номера станций, через которые будет проходить маршрут, на котором достигается такое значение L, в порядке следования. Гарантируется, что хотя бы один такой маршрут существует.

Пример

input.txt	*output.txt*
4 4 3 1 3 0.0 0.0 0.5 0.5 1.0 0.5 2.0 2.0 1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 0.0 1 2 2 3 3 4 1 4	0.5 1 2 3

Задача 7. Ролевая игра 2 (Division 1 only)

Ограничение по времени на 1 тест: ***2 сек.***

Вася очередной раз играет. Это ролевая игра, в которой предлагается на выбор несколько вариантов экипировки персонажа. Каждая экипировка имеет два параметра: сопротивляемость физическим и сопротивляемость магическим повреждениям. Для того чтобы оценить пригодность каждой экипировки, проводится некоторое испытание. Испытание состоит из последовательности воздействий. Производится три типа воздействий: физический урон, магический урон и лечение. При лечении количество здоровья персонажа увеличивается на заданную величину. При получении урона количество здоровья уменьшается на величину $D \cdot (1 - R)$, где D – заданный номинальный урон воздействия, а R – сопротивляемость экипировки соответствующему типу урона. Если в какой-то момент количество здоровья персонажа становится отрицательным, то он умирает. Начальное количество здоровья задано.

Требуется определить для каждой из заданных экипировок, переживет ли с ней Вася заданное испытание.

Входные данные

В первой строке входного файла записано три числа: целое N — количество воздействий в испытании, целое M — количество вариантов экипировки и вещественное H_0 — начальное количество здоровья ($1 \le N$, $M \le 100000$, $1.0 \le H_0 \le 1000000.0$). Следующие N строк содержат описания воздействия. Каждое воздействие записано в виде T_i D_i, где T_i — это один из символов **`H`**, **`P`**,**`M`**, обозначающий тип воздействия (**`H`** — лечение, **`P`** — получение физического урона, **`M`** — получение магического урона), а D_i – номинальный урон или количество вылеченного здоровья ($1.0 \le D_i \le 1000000.0$).

В последних M строках описаны варианты экипировок. Каждый вариант задан парой чисел: сопротивляемость физическому и сопротивляемость магическому уронам соответственно.

Все сопротивляемости и количества здоровья являются вещественными числами. Гарантируется, что суммарное количество вылеченного здоровья за всё испытание не превосходит 10^6. Аналогично, суммарный физический урон и суммарный магический урон каждый не превосходит 10^6.

Гарантируется, что если бы персонаж жил вечно (т.е. даже после того, как количество здоровья стало отрицательным), то минимальное количество здоровья на протяжении всего испытания в любой экипировке было бы либо меньше –0.0001, либо больше 0.0001.

Выходные данные

В выходной файл для каждой экипировки нужно вывести в отдельной строке слово **Alive**, если персонаж в ней выживет, и слово **Dead** в противном случае. Ответы для экипировок выводите в том же порядке, в котором они даны во входном файле.

Пример

input.txt	*output.txt*
5 6 10.0 P 6.0 H 4 M 10.0 H 15 P 10.0 0 0 0.5 0 0 0.12345 0.2 0.1 1 1 0.5 0.5	Dead Alive Dead Alive Alive Alive

Комментарий

Для четвертой экипировки из теста количество здоровья изменяется следующим образом:
10 → 5.2 → 9.2 → 0.2 → 15.2 → 7.2

Задача 8. Морской бой

Ограничение по времени на 1 тест: ***2 сек.***

Петя и Вася играют в морской бой на доске размером $M \times N$ клеток. После ожесточённой битвы у Пети остался лишь один двухпалубный корабль. Это прямоугольник размером 1×2, расположенный либо горизонтально, либо вертикально на поле. Вася не сомневается в своей победе. Осталось только отыскать его. Петя отметил на поле те клетки, в которых корабль стоять не может, и задумался: “Сколько же выстрелов нужно произвести, чтобы гарантированно «ранить» оставшийся корабль?” Помогите ему решить эту задачу.

Входные данные

В первой строке входного файла записаны два натуральных числа M и N – размеры поля для игры ($1 \le M, N \le 150$). Далее следует описание поля в следующем виде: M строк длины N из «`0`» и «`1`» («`1`» – клетка отмечена, «`0`» – нет). Гарантируется, что оставшийся корабль может поместиться на неотмеченных клетках.

Выходные данные

В первую строку выходного файла необходимо вывести одно натуральное число S – количество выстрелов, необходимых для того, чтобы «ранить» оставшийся корабль. В каждой из следующих S строк должно быть записано по паре натуральных чисел – координаты клеток поля, прострелив которые, Вася гарантированно «ранит» Петин корабль. Будем считать, что координаты правого нижнего угла поля – (M, N), а левого верхнего – (1, 1).

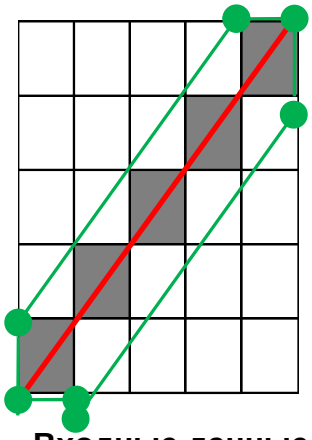
Пример

input.txt	*output.txt*
2 2 **00** **10**	**1** **1 2**

Задача 9. Растеризация отрезка (Division 1 only)

Ограничение по времени на 1 тест: **2 сек.**

Растровое изображение представляет собой прямоугольное клеточное поле размера M на N клеток (пикселей), изначально окрашенных в белый цвет. На растре введена прямоугольная система координат. Углы растра имеют координаты (0, 0), (M, 0), (M, N), (0, N). Ось X направлена вправо, а ось Y направлена вверх.



Через два противоположных угла с координатами (0, 0) и (M, N) проведена прямая линия. Все пиксели растра, которые она пересекает, полностью перекрашиваются в чёрный цвет. Получается некоторая фигура, состоящая из черных пикселей. Требуется найти выпуклую оболочку этой фигуры.

Пиксель перекрашивается в черный цвет, только если в пересечении прямой и пикселя получается отрезок. При касании пикселя и прямой пиксель не перекрашивается.

Входные данные

В первой строке входного файла задано целое число T — количество наборов входных данных ($1 \le T \le 1000$). Каждая из следующих T строк содержит по два целых числа M_i и N_i — размеры растра ($1 \le M_i, N_i \le 10^9$, $1 \le i \le T$).

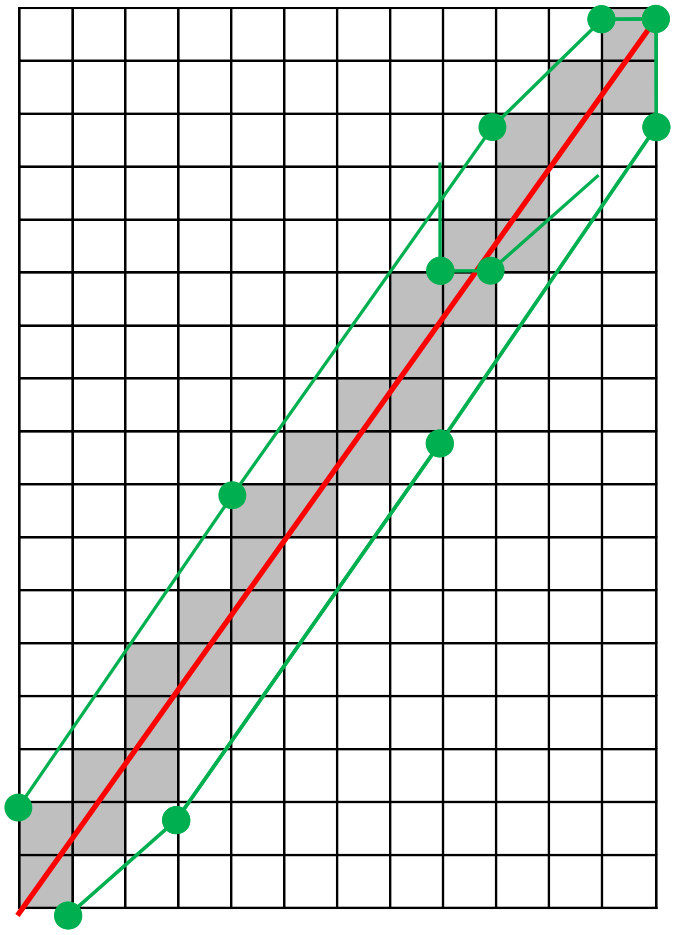
Выходные данные

Для каждого набора входных данных выведите искомую выпуклую оболочку в следующем формате. В первой строке должно быть записано целое число K_i — количество вершин выпуклой оболочки. В следующие K_i строк необходимо вывести по два числа — координаты вершин (сначала X-координата, потом Y-координата).

В выпуклой оболочке не должно быть трёх вершин, лежащих на одной прямой. Вершины требуется выводить в порядке обхода против часовой стрелки. Вершина (0, 0) должна быть выведена первой.

Пример

input.txt	*output.txt*
2 5 5 12 17	6 0 0 1 0 5 4 5 5 4 5 0 1 10 0 0 1 0 3 2 8 9 12 15 12 17 11 17 9 15 4 8 0 2



Задача 10. Ингредиенты

Ограничение по времени на 1 тест: ***2 сек.***

На заводе, производящем комбикорм для скота, делают порционный комбикорм для диетического питания овец. Каждая порция состоит из зерён N типов. Обозначим количество зёрен первого типа через C_1, второго типа – C_2 … количество зерен типа i – через C_i . Все зерна всех типов одинаковы по весу, так что вес порции оценивается числом зерён $S = \sum_{i=1}^{N} C_i$

Из шефствующего аграрного института поступило указание увеличить порции до веса T, сохранив процентное соотношение зерён как можно ближе к рецепту, подобранному ранее. Было решено оценивать «близость» нового рецепта по среднеквадратичному отклонению долей ингредиентов.

Например, если старый рецепт включал в себя всего по одному зернышку двух разных типов (т.е. рецепт 50% и 50%), а новый рецепт включает в себя три зернышка первого типа и одно зернышко второго типа (т.е. 75% и 25%), то среднеквадратичное отклонение долей составит:

$$\sqrt{\frac{(0{,}5-0{,}75)^2+(0{,}5-0{,}25)^2}{2}} = 0{,}25$$

Необходимо написать программу, которая вычисляла бы новый рецепт, в котором общий вес порции был равен T, и среднеквадратичное отклонение процентного содержания ингредиентов было минимальным.

В том случае, когда возможно несколько новых рецептов (например [1, 1, 1] → [2, 1, 1] или [1, 2, 1] или [1, 1, 2]), нужно выбрать тот, в котором наименьшее количество первого ингредиента. При фиксированном первом ингредиенте, аналогичное правило применяется ко второму и т.д.

Входные данные

В первой строке входного файла записаны через пробел два целых числа N и T – количество ингредиентов и целевой вес ($0 < N \le 100$, $0 < T \le 10^6$). На следующей строке через пробел записано N целых чисел C_i — количество зерен каждого типа в старом рецепте ($0 < C_i \le 1000$, $1 \le i \le N$).

Выходные данные

В первую строку выходного файла необходимо вывести число N. Во второй строке должны быть записаны через пробел N целых чисел, которые задают количество зерен каждого типа в новом рецепте.

Примеры

input.txt	*output.txt*
3 7 1 2 3	3 1 2 4
3 4 1 1 1	3 1 1 2

Задача 11. Трансляция (Division 1 only)

Ограничение по времени на 1 тест: ***2 сек.***

Вася Пупкин уверен — будущее за языком программирования, который нельзя называть. Он даже разработал процессор, способный напрямую исполнять его инструкции. Теперь он хочет подвергнуть коммерциализации свою разработку, и запустить процессор в производство. Но поскольку большинство программ написано на более “традиционных” языках, Васе нужны компиляторы, транслирующие программы в этот язык. Пока Вася решил ограничиться разработкой демонстрационного транслятора с другого очень простого языка, также придуманного им. Он назвал этот язык VSL – очень простой язык (Very Simple Language). В данный момент Вася очень занят созданием презентации для инвесторов. Поэтому написание транслятора он поручил вам.

Язык VSL, придуманный Васей, действительно, очень прост. Все переменные в нем имеют знаковый целочисленный тип размером 4 байта. Начальные значения всех переменных равны 0. Поскольку переменные имеют один и тот же тип, их предварительное объявление не требуется. В языке имеется всего 4 вида операторов: чтения, вывода, присваивания и сравнения. Все операторы, за исключением оператора сравнения, записываются на отдельной строке. Грамматика этих операторов представлена ниже.

operator ::= *read_op* | *write_op* | *assign_op*
read_op ::= **read(***var_name***)**
write_op ::= **write(***expression***)**
assign_op ::= *var_name* **=** *expression*
expression ::= *number* | *var_name* | *sum* | **(***expression***)**
sum ::= *expression* **+** *expression*

Здесь *number* — целочисленная неотрицательная константа, не превышающая 9, *var_name* — имя переменной, слово, состоящее только из маленьких латинских символов и цифр, причем цифра не может стоять в начале имени переменной. Каждая переменная имеет уникальное имя. Длина имени переменной не превышает 10 символов.

Оператор сравнения имеет следующий формат:

if(*expression cmp_op expression***)then**
operator1
operator2
……...
operatork
[**else**
operator1e
operator2e
……..
operatorme]
end

где *cmp_op* – один из символов ‘**<**’, ‘**>**’, ‘**=**’, ‘**#**’, означающих, соответственно, меньше, больше, равно и не равно. Конструкция **if** … **then** записывается на отдельной строке, далее идет некоторое, возможно нулевое, число строчек с операторами, которые следует выполнить в случае, если условие выполнено. Далее опционально следует **else** на отдельной строке, начиная блок операторов, выполняемых в случае, если условие не выполнено. Оператор заканчивается словом **end** на отдельной

строке. Слова **`read`**, **`write`**, **`if`**, **`then`**, **`else`** и **`end`** являются ключевыми и не могут быть именами переменных.

Вам нужно написать программу, которая транслирует программу, написанную на языке VSL, в программу на языке, который нельзя называть. Программа на языке, который нельзя называть, состоит из последовательности команд, исполняемых процессором, разработанным Васей. В системе помимо процессора имеется неограниченный объем памяти, состоящей из знаковых целочисленных 4-байтовых ячеек, а также указателя на текущую ячейку. Вся ячейки памяти в начальный момент установлены в нулевые значения. В языке, который нельзя называть, имеются следующие команды, каждая из которых выполняется за 1 такт:

Команда	Описание
`>`	Сдвигает указатель на одну ячейку вправо
`<`	Сдвигает указатель на одну ячейку влево
`+`	Увеличивает значение в ячейке, на которую указывает указатель, на 1
`-`	Уменьшает значение в ячейке, на которую указывает указатель, на 1
`[`	Если значение ячейки, на которую указывает указатель, равно нулю, переходит на парную данной команду **`]`** (с учетом вложенности), иначе продолжает выполнение команд со следующей
`]`	Если значение ячейки, на которую указывает указатель, не равно нулю, переходит на парную данной команду **`[`** (с учетом вложенности), иначе продолжает выполнение команд со следующей
`.`	Выводит значение ячейки, на которую указывает указатель, как байт. При этом ячейка должна иметь значение от 0 до 255 включительно.
`,`	Читает символ из потока ввода в ячейку, на которую указывает указатель. При этом ячейка принимает значение от 0 до 255 соответственно прочитанному символу.

Операторы открывающая и закрывающая квадратные скобки (‘**`[`**’ и ‘**`]`**’) являются парными. Это означает, что операторы ‘**`[`**’ и ‘**`]`**’ должны образовывать правильную скобочную последовательность. Формально правильные скобочные последовательности (ПСП) могут быть определены следующим образом:

ПСП ::= “” | **`[`**ПСП**`]`** | ПСП ПСП

Здесь через “” обозначена пустая строка, а ПСП ПСП означает конкатенацию двух, возможно различных, правильных скобочных последовательностей.

Заметьте, что поскольку память не ограничена, операторы **`>`** и **`<`** всегда могут быть выполнены. Поскольку переменные в языке являются знаковыми целыми, в результате операции уменьшения, примененной к числу 0, получается число –1. Программа на данном языке завершается, когда выполнение программы выходит за последнюю команду программы.

Для упрощения ввода-вывода будем считать, что все вводимые и выводимые числа больше либо равны 0 и не превосходят 9. На вход программе подается строка из символов — цифр, а оператор **`read()`** читает следующую непрочитанную цифру и записывает ее в переданную ему переменную. При этом оператор ‘**`,`**' языка, который нельзя называть, читает ASCII код следующей непрочитанной цифры в текущую ячейку. ASCII коды цифр ‘0’ … ‘9’ равны числам от 48 до 57 соответственно. Будем также полагать, что оператору **`write()`** передается выражение, значение которого лежит в диапазоне от 0 до 9, и оператор **`write()`** выводит единственный символ — цифру, равную значению выражения. Программа на языке, который нельзя называть, должна выдать строку символов с ASCII кодами от 48 до 57. Это означает, что при выполнении оператора «точка» в текущей ячейке должно находиться число от 48 до 57, соответствующее выводимой в данный момент цифре.

Входные данные

Во входном файле задана программа на языке VSL для трансляции. Гарантируется, что передаваемые программе на VSL на вход данные будут таковы, что значения любых вычисляемых в программе выражений (*expressions*) будут лежать в диапазоне от 0 до 9, при этом исходная программа

не будет читать более 4 цифр из входного потока. Количество строк в программе не превосходит 100. В каждое выражение (*expression*) входит не более чем 10 операций сложения. Пробелы могут стоять в произвольных местах между лексемами. Длина строк во входном файле не превосходит 256 символов.

Выходные данные

В выходной файл необходимо вывести единственную строку — программу на языке, который нельзя называть. Выданная программа считается корректной, если при любых входных данных, с учетом ограничений, указанных выше, эта программа выводит ту же самую строку цифр, что и исходная программа. При этом программа на языке, который нельзя называть, не должна пытаться прочесть больше цифр, чем исходная программа при таких же входных данных. Программа должна завершаться не более чем за 10^6 тактов. Количество команд в выходной программе также не должно превышать 10^6. Гарантируется, что такая программа существует.

Пример

input.txt	*output.txt*

`read (a)`
`if (a >0)then`
`read( b )`
`end`
`read(c)`
`write(a +(b) )`
	`,>++++++++[<------>-]<[<,<++++++++[>------`
	`<-]]>[>+<-]>>++++++++[<+++++++>-]<.`

Задача 12. Weather Balloon (Division 2 Only!)

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Ковбой Джо в течение длительного времени наблюдал за направлением ветра в прериях. В результате этих наблюдений он построил формулу, описывающую поведение метеорологического зонда. В частности, формула предсказывет высоту a (в метрах выше уровня земли) в момент t после запуска зонда.

$$A = -6t^4 + ht^3 + 2t^2 + t$$

При этом h — целое число между 1 и 100 включительно, зависящее от влажности воздуха.

Джо хочет дождаться момента, когда после запуска зонд коснётся земли. При этом, если зонд коснётся земли позже, чем через M часов, то считается, что Джо не удалось дождаться соответствующего момента. Примем, что зонд коснулся земли в случае, если $A \leqslant 0$.

Формат входного файла

Во входном файле заданы два неотрицательных целых числа: h — влажность воздуха, и M — максимальное время, в течение которого Джо готов ждать ($0 \leqslant h \leqslant 100$ and $0 < M < 240$).

Формат выходного файла

В случае, если зонд в первый раз коснётся земли позже, чем через N часов, выведите "`The balloon does not touch ground in the given time.`", в противном случае в первой строке выходного файла выведите "`The balloon first touches ground at hour:`", а во второй — целое положительное число T — наиболее раннее время, в которое высота зонда не будет превосходить 0.

Примеры

input.txt	output.txt
30 10	The balloon first touches ground at hour: 6
70 10	The balloon does not touch ground in the given time.

Задача 13. Blood (Division 2 Only!)

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

На станции переливания крови имеется некоторое количество крови всех четырёх групп: первой, второй, третьей и четвёртой. Также для крови определён "резус-фактор", который может быть как положительным, так и отрицательным.

Своей очереди на переливание крови ждёт некоторое количестов пациентов. При этом каждому пациенту требуется одна объёмная единица крови. Совместимость крови донора и пациента определяется следующими правилами:

- Пациентам с первой группой крови может быть перелита только кровь первой группы.
- Пациентам со второй группой крови может быть перелита кровь первой или второй группы.
- Пациентам с третьей группой крови может быть перелита кровь первой или третьей группы.
- Пациентам с четвёртой группой крови может быть перелита кровь любой группы.

Кроме того, пациентам с отрицательным резус-фактором может быть перелита только кровь с отрицательным резус-фактором. Пациентам с положительным резус-фактором может быть перелита кровь как с положительным, так и с отрицательным резус-фактором. Выясните, какому наибольшему количеству пациентов удастся сделать успешное переливание крови.

Формат входного файла

В первой строке входного файла содержатся 8 целых чисел — количество объёмных единиц крови первой группы с отрицательным резус-фактором, первой группы с положительным резус-фактором, второй группы с отрицательным резус-фактором, второй группы с положительным резус-фактором, третьей группы с отрицательным резус-фактором, третьей группы с положительным резус-фактором, четвёртой группы с отрицательным резус-фактором, четвёртой группы с положительным резус-фактором соответственно. Во второй в аналогичном формате задано количество пациентов с соответствующим сочетанием группы крови и резус-фактора. Все числа во входном файле неотрицательны и не превышают 10^7.

Формат выходного файла

Выведите одно целое число — наибольшее количество пациентов, которым удастся сделать успешное переливание крови.

Пример

input.txt	output.txt
5 5 3 1 2 11 5 12 2 4 9 2 3 9 7 3	33

Задача 14. Card Game (Division 2 Only!)

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Рассмотрим карточную игру для двух игроков. Карты в этой колоде делятся по четырём мастям (красная, жёлтая, зелёная или чёрная) и значениям (1, 2, 3, 4, 5). Всего в колоде содержатся 20 карт — каждая комбинация масти и значения представлена ровно один раз.

Каждому игроку раздаётся по 10 карт. Игра состоит из десяти раундов. Задача каждого игрока — выиграть как можно больше раундов. В каждом раунде определяется 'атакующий' игрок, который делает ход одной из карт, имеющихся у него на руках. Другой игрок должен сыграть картой той же масти, если таковая у него есть. Если нет, он сбрасывает любую карту, после чего обе карты выходят из игры на данный раунд. Атакующий выигрывает, если второй игрок не нашёл карты той же масти, или же если значение карты атакующего больше значения карты второго игрока. В противном случае выигрывает другой игрок.

Право атаки в первом раунде определяется жребием, в последующих девяти раундах атакует игрок, выигравший предыдущий раунд.

Выясните, сколько раундов выиграет игрок, атакующий первым, при том, что оба игрока придерживаются оптимальной стратегии.

Формат входного файла

Входной файл состоит из не более, чем 10 тестовых примеров. Каждый тестовый пример состоит из одной строки ввода, описывающей карты, которые сданы игроку, атакующему в первом раунде. Карты описываются парой «буква-цифра», где буква — одна из букв 'R','Y','G','B' соответственно для красной, жёлтой, зелёной и чёрной мастей, а цифра — одна из цифр (1,2,3,4,5), задающая значение карты. Остальные 10 карт розданы другому игроку.

После последнего тестового примера следует строка, содержащая 10 звёздочек, разделённых пробелами. Эту строку обрабатывать не следует.

Формат выходного файла

Для каждого тестового примера в отдельной строке выведите целое число — количество раундов, выигранных при оптимальной стратегии игроком, атакующим первым.

Пример

input.txt	output.txt
G1 G3 B2 R2 Y1 R3 R5 Y2 Y3 G5 * * * * * * * * * *	3

Задача 15. Multiple choice tests (Division 2 Only!)

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Школьный учитель Е.Г. Обломский часто даёт тесты, в которых требуется выбрать один правильный вариант из пяти. Е.Г. считает, что, во-первых, даже у не знающего предмет школьника остаётся интерес к тестам как к упражнениям на угадку, а, во-вторых, проверку таких тестов легко автоматизировать.

Ваша задача — написать систему автоматической проверки подобных тестов. Возможные ответы обозначаются буквами '`A`','`B`','`C`','`D`' or '`E`'.

Формат входного файла

В первой строке входного файла содержится целое число N ($0 < N < 10^4$) — количество заданий в тесте. В последующих N строках содержатся ответы ученика на соответствующие вопросы теста. Каждый ответ представляет собой один из символов '`A`', '`B`', '`C`','`D`' или '`E`'. Далее в аналогичном формате в N строках заданы правильные ответы, при этом вопросы перечислены в порядке их следования в тесте, то есть строка с номером i содержит ответ ученика, а строка с номером $N + i$ — правильный ответ на один и тот же вопрос.

Формат выходного файла

Выведите одно целое число — количество правильных ответов, данных учеником.

Примеры

input.txt	output.txt
3 A B C A C B	1
3 A A A A B A	2