Задача 1. Сложная документация

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Роман — технический писатель в крупной IT-компании «Текставей». Его работа заключается в написании различных документаций, правил, спецификаций. Объемы уже написанного текста немыслимо большие, и все было ничего, как в один момент приходит письмо сверху... В нем описывается как важно поддерживать информацию в документах актуальной и оформлять все в едином стиле, а также был прикреплен документ со сводом правил — описание этого единого стиля. Более того, была развернута целая система, которая проверяла все файлы с текстами и указывала на конкретные несоответствия.

Роман сразу же побежал смотреть отчет по его документам и увидел порядка 100500 проблем, которые срочно нужно исправить. Его расстройству не было предела, но он мужественно решил проанализировать, что же там не так. Выяснилось, что основная часть несоответствий — это одно правило, и звучит оно следующим образом: «Перед и после каждого двоеточия и тире должен быть хотя бы один пробел или начало/конец строки».

В этот же момент Роман подумал, что можно изобрести программу, которая исправит его документы, хотя бы для этого правила. Программа должна добавлять в текст минимальное количество пробелов так, чтобы правило выполнялось.

К сожалению, Роман технический писатель и не умеет программировать. Помогите ему в этом нелегком вопросе.

Формат входных данных

В первой строке входного файла записано единственное целое число T — количество строк в тексте ($1 \leqslant T \leqslant 10\,000$). Далее следует T строк — текст документа.

Гарантируется, что суммарное количество символов в тексте не превышает 10 000. Все символы имеют ASCII-коды от 32 до 126 включительно.

Формат выходных данных

Выведите в выходной файл исправленный текст.

Примеры

input.txt
2 This document describes a secret: new powerful product-Arkana! What is that? Let us start:
output.txt
This document describes a secret : new powerful product - Arkana! What is that? Let us start :

Комментарий

Ни одна строка в примере не заканчивается пробелом.

Задача 2. Антиталия

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

Выдающиеся сибирские учёные получили шнобелевскую премию по анатомии за статью «Талия и антиталия». Как это всегда бывает с серьёзными достижениями, новая идея стала проникать в другие области науки и техники. Оказывается, понятие антиталии полезно и в инженерном деле: оно позволяет выявлять наиболее прочные части конструкции. Одна беда, у инженеров нет программы, которая могла бы найти антиталию у твёрдого тела.

Рассмотрим твёрдое тело. Можно построить сечение этого тела любой горизонтальной плоскостью $z = \text{const}$. Антиталией называется горизонтальное сечение, которое имеет наибольшую возможную площадь.

Твёрдое тело — это регулярное множество точек в трёхмерном пространстве. В теле не может быть бесконечно тонких элементов. Тело задаётся своей границей, которая представлена в виде триангуляции.

Триангуляция дана в индексированном виде, как массив вершин и массив треугольников. Все вершины различны и расположены в точках с целочисленными координатами. Каждый треугольник задан индексами трёх его вершин в общем массиве вершин. Все три индекса различные, площадь треугольника не равна нулю. Порядок индексов — против часовой стрелки, если смотреть на треугольник извне тела.

Поскольку триангуляция ограничивает твёрдое тело, в ней нет самопересечений и самокасаний. Два треугольника могут иметь общий отрезок, если у них две вершины совпадают, и общую точку, если у них одна вершина совпадает — других общих точек нет. Немanifолдных рёбер и вершин нет, то есть, каждое ребро является стороной двух треугольников; множество треугольников, содержащих любую вершину, сцеплено по своим сторонам в один цикл. Тело может быть несвязным, а также содержать полости.

Формат входных данных

В первой строке входного файла записано два целых числа: V — количество вершин и T — количество треугольников ($4 \leqslant V, T \leqslant 200\,000$). В следующих V строках описываются вершины. В каждой строке три целых числа — x, y и z координаты вершины соответственно. В оставшихся T строках описываются треугольники. В каждой строке записано по три целых числа — индексы вершин треугольника. Вершины нумеруются с нуля в порядке их описания.

Координаты не превышают 500 000 по абсолютной величине. Используется правосторонняя система координат.

Формат выходных данных

В выходной файл необходимо вывести найденную антиталию. Она описывается на одной строке двумя вещественными числами: z-координатой плоского сечения и площадью этого сечения. Если сечений с наибольшей площадью несколько, необходимо вывести z-координату любого из них.

Рекомендуется выводить вещественные числа с максимальной точностью. Абсолютная или относительная ошибка площади выведенного вами сечения не должна превышать 10^{-8}.

Примеры

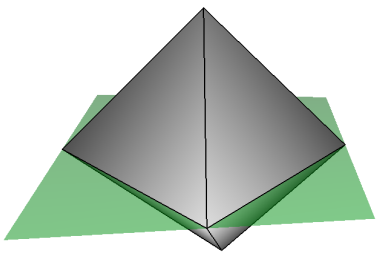
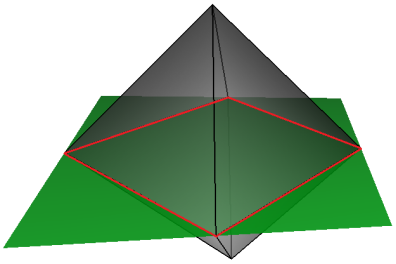
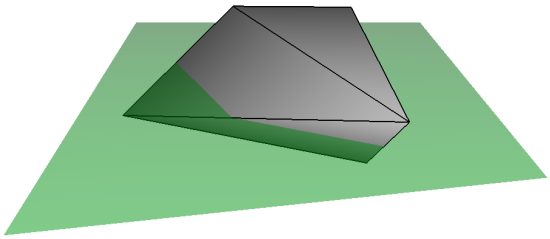
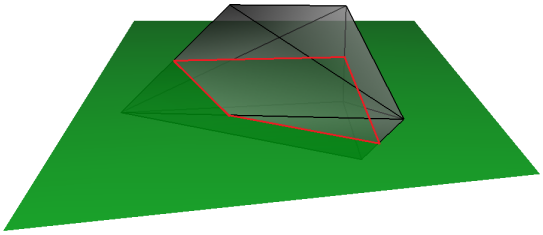
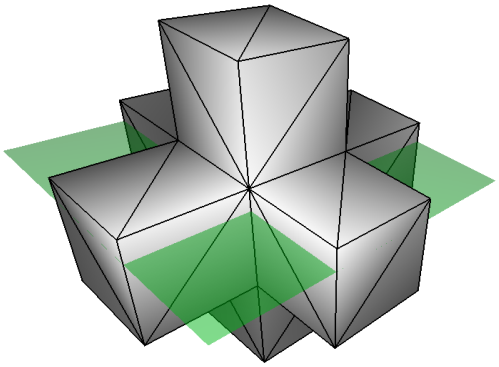
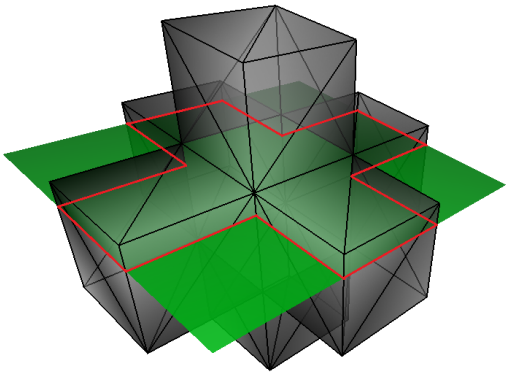
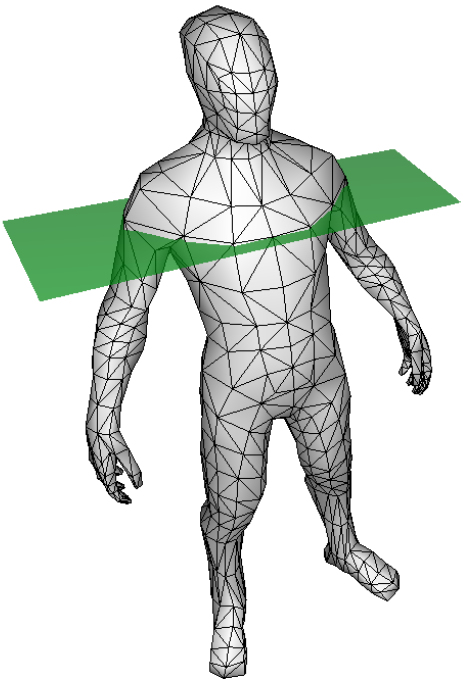
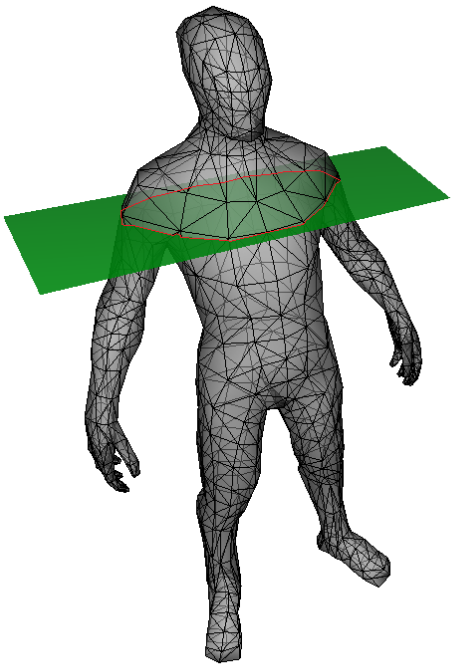
Входные данные для последних двух примеров в таблице **не** показаны. Их можно найти в архиве примеров, который можно скачать рядом с этими условиями задач.

input.txt	output.txt
6 8 1 0 0 -1 0 0 0 1 0 0 -1 0 0 0 1 0 0 -1 2 0 5 1 4 2 0 2 4 5 0 3 1 3 4 3 1 5 2 5 1 4 3 0	0 2
6 8 0 0 0 2 0 0 0 1 0 0 0 1 1 0 1 0 2 1 1 0 2 4 5 3 0 1 3 3 1 4 3 5 0 5 2 0 4 1 5 5 1 2	0.5 1.5
32 60 ... Полные данные в архиве примеров!	0.7913756716346761 5
896 1788 ... Полные данные в архиве примеров!	-3318.6080203949518 4003775.6378878844

Пояснение к примеру

В первом примере тело является правильным октаэдром с центром в начале координат, все вершины лежат на осях координат. Самое большое сечение — это среднее сечение $z = 0$, в котором получается квадрат со стороной $\sqrt{2}$. Во втором примере максимальное сечение также достигается на средней высоте. Во третьем примере дан крест из шести кубических выступов. Наибольшее сечение получается при $0 \leqslant z \leqslant 1$ — получается плоский крест. В последнем примере тело является телом человека. Можно увидеть, где обычно у человека антиталия.

На следующей странице приведены иллюстрации для примеров.

Задача 3. Найти радистку

Имя входного файла: `стандартный поток ввода`
Имя выходного файла: `стандартный поток вывода`
Ограничение по времени:1 секунда
Ограничение по памяти: 256 мегабайт

Штирлицу на парашюте сбросили новую радистку взамен старой. Но вот только промахнулись, и та приземлилась неизвестно куда. Хорошо хоть, что на плоскость и в точку с целыми координатами. Ну, деваться некуда, придется разыскивать барышню... Хоть и не полковничье это дело — самому бегать, шариться по буеракам, но поручить его некому. Один зам в отпуске, другой в загуле, а третий, хоть и вышел и из отпуска и из загула, но ни на что не способен. Для того, чтобы справиться с заданием, Штирлиц по старой дружбе одолжил у герра Шелленберга машинку для поимки радисток и ездит по полю, пеленгует страдалицу.

Принцип работы новейшего аппарата, очередного гениального изобретения слесаря-самоучки Полесова, следующий: в некоторой точке плоскости машина останавливается, направляет антенну в некотором направлении и измеряет уровень сигнала. Если Штирлиц находится в той же точке, что и радистка, то машина выдаёт значение -1 независимо от направления. В противном случае машина показывает уровень сигнала $P\cos\varphi/R^2$, если $\varphi < 90$ градусов, и ноль, если $\varphi \geqslant 90$ градусов. Здесь φ — это угол между вектором направления антенны и вектором, направленным из машины Штирлица на радистку, R — расстояние от машины до радистки, а P — неизвестная константа, зависящая от радиооборудования.

Задача Штирлица: определить место нахождения радистки не более, чем за 10 измерений-пеленгов, а то прибегут ребята из конкурирующей конторы Мюллера.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Чтобы сделать запрос, нужно вывести в стандартный поток вывода знак вопроса и четыре целых числа x_0, y_0, x_d, x_d, разделенные пробелом, где x_0, y_0 — координаты машины Штирлица, а x_d, y_d — компоненты вектора направления антенны ($|x_0|, |y_0|, |x_d|, |y_d| \leqslant 10^4$). Вектор направления должен быть ненулевым, т.е. $|x_d| + |y_d| > 0$.

В ответ на запрос в стандартный поток ввода приходит одно вещественное число: мощность сигнала, показанная машиной. Если это значение равно -1, то следует сразу завершить работу программы, ведь радистка уже найдена.

Координаты радистки целые и не превышают 10^3 по абсолютной величине. Константа P вещественная, лежит в пределах от 1 до 10^6.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного запроса. Иначе решение может получить вердикт Timeout.

Пример

Для удобства чтения команды в примере разделены строками с символом минуса. Ваша программа **не** должна выводить никаких минусов.

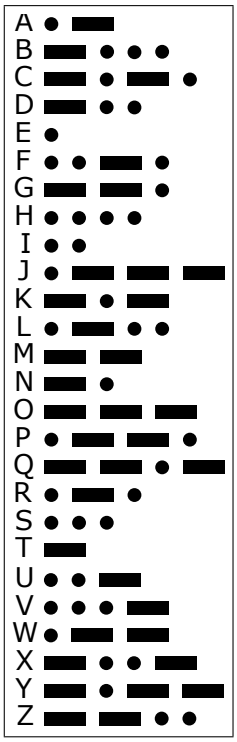
стандартный поток ввода	стандартный поток вывода
- 0.035999999999999972799536 - 0.000000000000000000000000 - 0.008999999999999993199884 - 0.089999999999999827915431 - 0.166666666666666574148081 - -1	? 0 0 0 1 - ? -2 -3 -1 1 - ? -4 -3 0 100 - ? 3 0 3 1 - ? 4 0 0 1 - ? 4 3 1 1

Пояснение к примеру

В примере координаты радистки (4, 3), а константа $P = 1.5$. Как видно, интерактор вычисляет мощность сигнала с помощью типа double и выводит её с большим количеством знаков после десятичной точки.

Задача 4. Код Морзе

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт



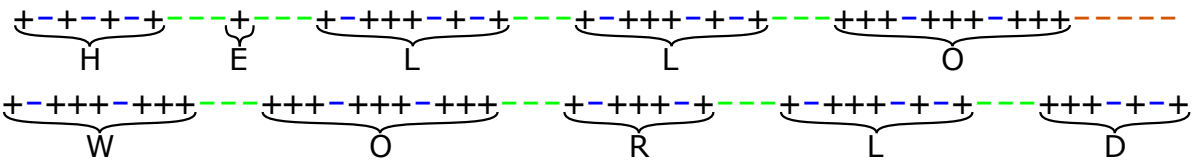
В азбуке Морзе каждая буква представляется последовательностью «точек» и «тире». Однако одного лишь этого недостаточно, чтобы передавать текст по радио. Помимо этого, код Морзе предписывает длительности для точки и тире, а также для различных пауз между ними.

В данной задаче будет рассматриваться уже оцифрованный радиосигнал, закодированный на той стороне с помощью кода Морзе. Этот сигнал представляется строкой из символов «плюс» и «минус». Каждый символ определяет, был ли звук в соответствующий такт передачи: плюс означает, что звук был, а минус — что звука не было.

В рамках данной задачи будем считать, что в коде Морзе есть элементы пяти типов. Эти элементы записываются в сигнал согласно таблице:

`+`	«точка»	есть звук в течение одного такта
`+++`	«тире»	есть звук три такта
`-`	пауза внутри буквы	нет звука один такт
`---`	пауза между буквами	нет звука три такта
`-----`	пауза между словами	нет звука **пять** тактов

Например, если пользоваться стандартной азбукой Морзе для латинского алфавита, то два слова `HELLO WORLD` будут закодированы в сигнал из 63 элементов и 109 тактов:



Обратите внимание, что между словами стоит пятитактовая пауза, а в начале и в конце текста нет никаких пауз и специальных элементов.

Процедура автоматической оцифровки иногда ошибается, из-за чего расшифровать сигнал становится очень непросто. Процедура может укоротить или удлинить любой элемент на один такт, за исключением того, что однотактовый элемент не может стать короче.

Вам дан словарь и оцифрованный сигнал. Требуется определить, какое минимальное количество ошибок могла допустить процедура оцифровки, а также необходимо восстановить исходный текст при условии, что:

1. исходный текст состоял только из тех слов, которые есть в словаре,
2. используется абзука Морзе латинского алфавита,
3. не происходят никакие другие ошибки, кроме тех, которые описаны выше.

Формат входных данных

В первой строке входного файла записано два целых числа: M — количество слов в словаре и N — количество тактов в сигнале ($1 \leqslant M, N \leqslant 5\,000$).

В следующих M строках записаны слова в словаре, по одному в строке. Каждое слово непустое и состоит исключительно из заглавных букв латинского алфавита. Сумма длин всех этих слов не превышает 5 000.

В оставшихся строках записан оцифрованный сигнал — последовательность из символов «плюс» и «минус» общей длины N. Последовательность может быть разбита произвольным образом на несколько непустых строк. В сигнале есть хотя бы один «плюс».

Азбуку Морзе латинского алфавита в текстовом виде можно скачать там же, где можно скачать эти условия задач.

Формат выходных данных

Если указанный во входных данных сигнал нельзя получить с соблюдением всех условий задачи, то в выходной файл необходимо ввести одно число `-1`. Иначе, в первую строку требуется вывести одно целое число $K \geqslant 0$ — минимально возможное количество ошибок, а во вторую строку — исходный текст, из которого мог получиться сигнал с таким количеством ошибок. Текст следует выводить, разделяя соседние слова одним пробелом.

Если существует несколько вариантов исходного текста, из которых можно получить заданный сигнал с минимальным количеством ошибок, то необходимо вывести **лексикографически минимальный** из них. Считается, что пробел меньше любой буквы.

Примеры

input.txt	output.txt
2 109 HELLO WORLD +-+-+-+---+---+-+++-+-+--- +-+++-+-+---+++-+++-+++----- +-+++-+++---+++-+++-+++--- +-+++-+---+-+++-+-+---+++-+-+	0 HELLO WORLD
3 115 WORLD PROGRAM HELLO +-+-+-+---++--+-++++-+-+--- +-++-+-++----+++-+++-+++------ +-+++-+++---+++-+++-++--- +-++++-+---++-+++-+-+---+++-+--++	12 HELLO WORLD
1 13 E +-----------+	-1
1 13 HELLO +--+--+--+--+	-1

Пояснение к примеру

В первом примере записан пример из условия без каких-либо ошибок. Во втором примере текст тот же, но двенадцать элементов укорочено/удлинено на такт.

В третьем примере есть 11-тактовая пауза, которую никак получить нельзя. Самый длинный элемент — это 5-тактовая пауза между словами, которая может стать 6-тактовой в результате ошибки.

В четвёртом примере есть пять точек, и они могут составлять буквы `E`, `I`, `S` и `H` в разных комбинациях, однако в словаре есть лишь слово `HELLO`, которое из них никак не составить.

Задача 5. Леспромхоз

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

> *— Слушайте, ребята! Вот представим, что это — делянка. Валим самый большой ствол, а эти кладём все на него. Собираем это в чокер, и ты сразу всё это волокёшь. Понятно?*
> *— А что, толково! Молодняк сохраняем. Пеньки не мешают. Троса не рвём. Производительность во! И зарплата...*
>
> Фильм «Девчата»

Бригада лесорубов Ильи Ковригина придумала новый подход, как повысить производительность труда. Общая идея в том, чтобы несколько деревьев валить примерно в одно место, так чтобы можно было их все привязать за один раз к трактору и все вместе увезти. В данной задаче требуется определить, какое максимальное количество деревьев можно повалить так, чтобы все их можно было привязать к неподвижному трактору.

Будем считать, что земля — это горизонтальная плоскость. Из земли растёт N деревьев, каждое дерево — это направленный отрезок, у которого есть начало и конец. Растущее дерево представляется вертикальным отрезком, начало которого находится точно на земле. Для каждого дерева указаны его координаты (x_i, y_i) и высота h_i.

Если дерево свалить, то оно станет отрезком в плоскости земли, начало которого находится в той же точке (x_i, y_i), и длина которого по-прежнему равна h_i. Направление от начала сваленного дерева к концу может быть произвольным, и его выбирает работник, который валит дерево.

Трактор можно пригнать в любую точку. Привязать к трактору можно только те сваленные деревья, концы которых находятся на расстоянии не более R от трактора. Будем считать, что деревья и трактор никак не мешают друг другу.

Формат входных данных

В первой строке входного файла записано целое число N — количество деревьев на делянке, и вещественное число R — радиус охвата трактора ($1 \leqslant N \leqslant 250$, $\frac{1}{10} \leqslant R \leqslant 1\,000$). В остальных N строках описаны деревья. Каждое дерево описывается тремя вещественными числами: координатами x_i, y_i на плоскости и высотой h_i ($|x_i|, |y_i| \leqslant 1\,000$, $\frac{1}{10} \leqslant h_i \leqslant 1\,000$).

Все вещественные числа заданы с не более чем 15 знаками после десятичной точки. Гарантируется, что никакие два дерева не стоят на расстоянии меньше $\frac{1}{10}$ друг от друга. Гарантируется, что если увеличить радиус захвата R на 10^{-3}, то максимальное количество деревьев в ответе не изменится.

Формат выходных данных

В первую строку выходного файла необходимо вывести одно целое число A — какое максимальное количество деревьев можно привязать к трактору за раз. Во вторую строку выведите два вещественных числа X^* и Y^* — координаты точки, куда нужно поставить трактор. В остальных A строках должно быть записано, как и какие деревья валить. В каждую из этих строк выведите целое число k_i — номер дерева, которое надо свалить, и φ_i — полярный угол, определяющий направление валки дерева ($1 \leqslant k_i \leqslant N$, $0 \leqslant \varphi_i \leqslant 360$).

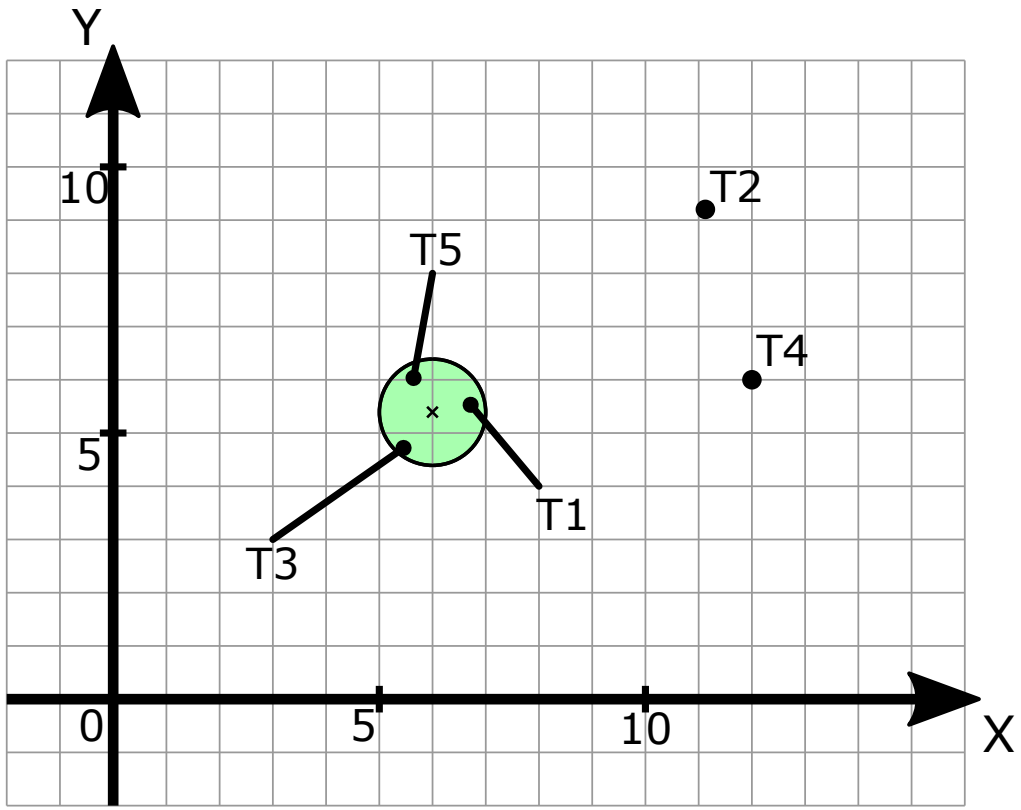
Деревья нумеруются в порядке описания во входном файле, начиная с единицы. Порядок вывода деревьев значения не имеет. Полярный угол измеряется начиная от оси X в направлении оси Y, выводится в градусах, и определяет направление от начала сваленного дерева к его концу.

Рекомендуется выводить все вещественные числа с 15 знаками после десятичной точки. Проверяющая программа будет проверять ваше решение с радиусом R, увеличенным на 10^{-6}.

Примеры

input.txt	output.txt
5 1.0 8.0 4.0 2.0 11.13 9.19 0.55 3 3 3 12 6 1 6 8 2	3 6 5.41234567890 1 130 5 259.726501937845610 3 35

Иллюстрация



Задача 6. Кеширование приближений

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
4 секунды (для Java)
Ограничение по памяти: 256 мегабайт

В геометрическом моделировании высокой точности нужно уметь хранить сложные кривые, такие как, например, кривая пересечения двух гладких поверхностей. Сложность заключается в том, что такие кривые практически невозможно представить в одновременно максимально точном и удобном для работы виде: либо представление получается приближённым, либо оно настолько сложное, что его нельзя использовать в половине геометрических алгоритмов без их полного переписывания. Поэтому на практике используют смешанный подход: для сохранения максимальной точности кривую хранят в сложном виде, но в алгоритмы передают приближение кривой простого вида, построенное с достаточной точностью.

На практике лучше всего зарекомендовало себя приближение кривой кубическим сплайном Эрмита — его легко строить и с ним легко работать. К сожалению, чем лучше точность приближения, тем больше памяти оно занимает, тем дольше оно строится, и тем медленнее на нём работают любые алгоритмы. Приближение кубическим сплайном имеет четвёртый порядок ошибки, то есть при улучшении точности в 16 раз размер приближения увеличивается в 2 раза.

В данном задаче мы будем считать, что приближение с точностью ε имеет размер M байтов в памяти, вычисляющийся по формуле:

$$M = \frac{s}{\sqrt[4]{\varepsilon}}$$

Для построения этого приближения требуется потратить T тактов процессорного времени:

$$T = aM + b$$

Здесь s, a и b — некоторые заданные константы.

Чтобы не перевычислять много раз одно и то же приближение для сложной кривой, его иногда «кешируют», то есть сохраняют рядом с кривой, чтобы использовать его многократно в дальнейшем. В данной задаче имеется одна сложная кривая, и от вас требуется определить, как лучше кешировать её приближения, чтобы как можно быстрее выполнить над ней заданные N операций.

Выполнять операции нужно строго последовательно, в том порядке, в котором они заданы. Ни одну операцию нельзя выполнить напрямую, передав в неё сложную кривую, в каждую операцию нужно обязательно передать подходящее приближение кривой. У каждой операции указана «толерантность» δ — требуемый порог точности. Приближение кривой, построенное с точностью ε подходит для операции с толерантностью δ, только если $\varepsilon \leqslant \delta$.

Время выполнения операции определяется по формуле:

$$T = cM + d$$

Здесь c и d — некоторые константы, заданные отдельно для каждой операции, а M — это размер используемого приближения кривой. Напомним, что этот размер определяется по точности построения ε согласно формуле, указанной выше.

Требуется определить, за какое минимальное время можно выполнить все операции при трёх различных политиках кеширования приближений:

1. **Кеширование отключено**: нет никакой возможности сохранять приближения кривой между операциями. После завершения операции используемое приближение сразу выбрасывается, и для следующей операции необходимо строить новое приближение.

2. **Кеширование одного приближения**: каждый раз, когда строится приближение кривой, оно сохраняется в кеше, вытесняя оттуда старое приближение, если оно имеется. Сохранённое в кеше приближение можно передавать в будущие операции сколько угодно раз, до тех пор, пока не будет построено новое приближение. Заметим, что вытеснение старого приближения происходит всегда, даже если новое приближение менее точное — нельзя построить приближение, не сохранив его в кеш.

3. **Неограниченное кеширование**: все построенные приближения сохраняются в кеше в неограниченном количестве и доступны для использования в будущем. Для каждой операции можно выбрать любое из ранее построенных приближений, конечно при условии, что оно подходит для операции.

Во всех трёх вариантах изначально кеш пуст, никаких приближений нет. Перед каждой операцией разрешается построить приближение с любой желаемой точностью $\varepsilon > 0$, либо ничего не строить. Время построения приближений добавляется к общему времени выполнения операций.

Формат входных данных

В первой строке входного файла записано целое число N — количество операций, которые надо выполнить ($1 \leqslant N \leqslant 10\,000$). Во второй строке записано три вещественных числа s, a и b — константы, влияющие на размер и время построения приближения ($10^{-3} \leqslant s \leqslant 10^3$, $10^{-4} \leqslant a, b \leqslant 10^4$).

В оставшихся N строках описаны операции. В каждой строке указано три вещественных числа: δ — требуемая точность операции, а также c, d — константы, влияющие на время выполнения операции ($10^{-12} \leqslant \delta \leqslant 1$, $10^{-4} \leqslant c, d \leqslant 10^4$).

Гарантируется, что толерантность δ у любых двух операций либо точно совпадает, либо отличается хотя бы на 10^{-3} в относительном смысле. Вещественные числа могут быть заданы в экспоненциальном виде, например: `1e-10`

Формат выходных данных

В выходном файле должно быть записано три ответа, каждый ответ состоит из $N + 1$ строки. Между ответами требуется вывести строку, состоящую из трёх знаков «равно».

Первый ответ получается, если кеширование отключено, второй — если кешируется одно приближение, и третий — если кешируются все приближения.

Ответ начинается со строки с вещественным числом τ, равным суммарному времени построения всех приближений и выполнения всех операций. В каждой i-ой из оставшихся N строк должно быть указано, нужно ли строить приближение кривой перед i-ой операцией, и если нужно, то какое ($1 \leqslant i \leqslant N$).

Если приближение строить не надо, необходимо вывести в строку -1, а если надо — вещественное число ε, показывающее, с какой точностью строить приближение.

При проверке решения будет учитываться требование по точности приближения **с относительной** точностью 10^{-12} и суммарное время работы с относительной точностью 10^{-8}. Рекомендуется выводить все вещественные числа **в экспоненциальном виде** с 15 знаками после десятичной точки.

Примеры

input.txt	output.txt
1 1.12e-1 0.25 1.37 1.2345e-3 0.57e+2 37.019	72.59645309369008839670076843792 8 1.2345e-3 === 7.259645309369008e+1 0.12345e-2 === 0.7259645309369008e+2 0.0012345
4 1 2 1 1e-4 1e-3 1 1e-12 1 1 1e-8 1e+3 1 0.0625e-8 0.1 1	103648.01 1e-4 1e-12 1e-8 0.0625e-8 === 103628 1e-12 -1 1e-8 0.0625e-8 === 103306.1 1e-08 1e-12 -1 -1

Пояснение к примеру

В первом примере только одна операция, поэтому независимо от политики кеширования ответ одинаковый, в любом случае нужно построить приближение для кривой с точностью, равной толерантности операции. Заметим, что размер приближения может быть нецелым, никаких округлений не делается. В выходном файле показаны разные способы вывода ответа.

Во втором примере при кешировании одного приближения выгодно построить сразу приближение с точностью 10^{-12} и использовать его в первых двух операциях, потому что время выполнения первой операции слабо зависит от размера используемого приближения, а значит, передать более точное приближение будет дешевле, чем строить ещё одно приближение меньшего размера. Третья операция очень сильно зависит от размера приближения, поэтому для неё приближение лучше перестроить с максимальным ухудшением точности. Однако такое приближение не подойдёт для четвёртой операции, поэтому для неё придётся построить несколько более точное приближение.

С другой стороны, при политике неограниченного кеширования выгоднее использовать для четвёртой операции изначально построенное приближение с точностью 10^{-12}. Кроме того, можно приближение с точностью 10^{-8} построить раньше, чтобы использовать его для первой операции.

Задача 7. Планировщик

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

В многозадачной операционной системе «Белочка ОС» запущено T процессов. Для каждого из T процессов задан приоритет p_i, который влияет на то, как часто выполняется процесс. Поскольку в распоряжении системы есть лишь одно ядро одного процессора, то перед операционной системой стоит непростая задача распределения процессорного времени между данными процессами, учитывая приоритеты процессов.

Алгоритм определения того, какой из процессов выполняется в каждый момент времени, можно описать следующим образом. Для каждого процесса помимо приоритета p_i поддерживается счётчик t_i. Изначально все t_i равны 0. Далее каждую секунду:

1. Выбираются процессы с максимальным значением $p_i + t_i$.
2. Среди таких процессов выбирается процесс с минимальным номером i.
3. Выбранный процесс i выполняется в течение секунды.
4. Для выбранного процесса i значение t_i устанавливается равным 0.
5. Для всех остальных процессов значение t_i увеличивается на 1.

Промоделируйте работу операционной системы в течение T секунд и вычислите, сколько секунд выполнялся каждый из процессов. Можно считать, что все вычисления и переключения процессов система выполняет мгновенно, поэтому время работы любого процесса в секундах будет целым числом.

Формат входных данных

В первой строке через пробел записаны два целых числа N и T — количество процессов в операционной системе ($1 \leqslant N \leqslant 10^5$) и сколько секунд надо промоделировать ($1 \leqslant T \leqslant 10^6$).

Во второй строке через пробел записаны N целых чисел p_i — приоритеты процессов ($0 \leqslant p_i \leqslant 10^5$).

Формат выходных данных

В единственной строке выходного файла выведите через пробел N целых чисел — сколько секунд выполнялся каждый из процессов.

Примеры

input.txt	output.txt
3 10 3 4 5	3 3 4

Задача 8. Текстовый редактор

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 3 секунды
5 секунд (для Java)
Ограничение по памяти: 256 мегабайт

При использовании простого текстового редактора иногда возникает необходимость поменять порядок строк определённым образом. Если строки длинные и хочется свести к минимуму вероятность ошибки, то проще всего сделать это, переставляя куски текста с места на место (метод «вырезать/вставить»). Вы — эффективный программист. Определите, за какое минимальное время можно переставить строки требуемым образом.

В файле N строк текста. В редакторе имеется один курсор, для которого возможно $N+1$ положение: до первой строки текста, после последней строки текста, и между любыми двумя соседними строками текста. Нажатие клавиши «вверх» или «вниз» перемещает курсор на одну строку выше или ниже соответственно. За пределы текста переместить курсор нельзя.

Клавиша Shift отвечает за выделение набора строк. Когда её нажимают, редактор запоминает текущую позицию курсора, а когда отпускают — выделяет все строки между текущей позицией и запомненной. После отпускания Shift-а необходимо нажать Ctrl+X — тогда все выделенные строки будут скопированы в буфер обмена и одновременно удалены из текста. Для того, чтобы вставить вырезанный кусок текста, нужно нажать Ctrl+V, переместив предварительно курсор в нужное место с помощью клавиш «вверх» и «вниз». После вставки текст пропадает из буфера обмена. При вырезании куска текста все строки после него автоматически смещаются вверх, а при вставке — смещаются вниз, так что в тексте никогда нет пустых строк. При вырезании курсор перемещается вверх в строку, с которой начинался вырезанный кусок, если только он уже там не находится, а при вставке курсор перемещается вниз, и оказывается сразу после вставленного куска текста.

Каждое действие отнимает определённое время, которое зависит от навыков пользователя. Изначально курсор находится перед первой строкой текста.

Формат входных данных

В первой строке входного файла записано одно целое число N — количество строк в текстовом редакторе ($2 \leqslant N \leqslant 8$). Во второй строке указано шесть целых чисел t_i — сколько миллисекунд занимает каждое действие ($1 \leqslant t_i \leqslant 100$). Список действий приведён ниже.

В третьей строке N целых чисел определяют, в каком порядке идут строки изначально. Эти числа различны и лежат в диапазоне от 1 до N. В четвёртой строке N чисел определяют, в каком порядке строки нужно расположить. Эти числа также различны и лежат в диапазоне от 1 до N.

Формат выходных данных

В первую строку необходимо вывести два целых числа: T — суммарное время выполнения плана в миллисекундах и K — количество действий в нём. Далее выведите K действий в порядке их выполнения, по одному в строке. Действия описываются следующим образом:

- `Up` — нажать клавишу «вверх» (требует t_1 миллисекунд).
- `Down` — нажать клавишу «вниз» (требует t_2 миллисекунд).
- `Shift-Press` — зажать клавишу Shift (требует t_3 миллисекунд).
- `Shift-Release` — отжать клавишу Shift (требует t_4 миллисекунд).
- `Ctrl+X` — нажать комбинацию Ctrl+X (требует t_5 миллисекунд).
- `Ctrl+V` — нажать комбинацию Ctrl+V (требует t_6 миллисекунд).

Пример

input.txt	output.txt
6 99 98 100 97 99 98 1 2 3 4 5 6 6 5 4 3 2 1	3252 33 Shift-Press Down Down Shift-Release Ctrl+X Down Down Ctrl+V Shift-Press Down Shift-Release Ctrl+X Down Ctrl+V Shift-Press Up Up Up Shift-Release Ctrl+X Up Up Ctrl+V Down Shift-Press Up Up Up Shift-Release Ctrl+X Up Up Ctrl+V

Пояснение к примеру

В примере нужно развернуть шесть строк, каждое действие отнимает 100 миллисекунд или чуть меньше. Предложенный план действий записан в gif-картинку, которую можно скачать там, где вы скачали условия задач.

Задача 9. Решающий удар

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

Ходислав играет в ролевую игру «Подземелья и драконы». В данный момент его персонаж сражается с монстром, и Ходислав, по какой-то причине уверенный в атаке, хочет уничтожить врага последним решающим ударом. У персонажа есть разное оружие, урон каждого оружия определяется броском игральных костей и характеризуется тремя числами n, f и m, где n — количество костей, f — количество граней каждой кости, m — модификатор. Например, если $n = 3$, $f = 8$, $m = 5$, то для определения урона следует бросить три восьмигранные кости, сложить результаты и добавить пять к сумме; обычно это записывается в виде $3d8 + 5$.

Чтобы уничтожить монстра, оружие должно нанести урон D или более. Помогите Ходиславу выбрать то оружие для персонажа, которое сделает это с наибольшей вероятностью.

Броски костей независимы, каждая грань кости выпадает равновероятно. На гранях кости написаны все числа от 1 до f.

Формат входных данных

В первой строке входного файла записано единственное целое число T — количество тестов ($1 \leqslant T \leqslant 5\,000$). Далее следует описание T тестов.

В первой строке теста записано два целых числа: W — количество оружия у персонажа и D — минимально необходимый урон монстру ($1 \leqslant W \leqslant 5\,000$, $1 \leqslant D \leqslant 250$).

В оставшихся W строках дано описание оружия. В каждой строке указано три целых числа: n — количество костей, f — количество граней каждой кости и m — модификатор ($1 \leqslant n \leqslant 10$, $2 \leqslant f \leqslant 20$, $-10 \leqslant m \leqslant 10$).

Гарантируется, что суммарное количество оружия по всем тестам не превышает 5 000.

Формат выходных данных

Для каждого теста в отдельную строку необходимо вывести одно вещественное число — максимальную вероятность нанести не меньше D урона одним ударом оружия. Ответ требуется вывести с абсолютной погрешностью не больше 10^{-11}.

Примеры

input.txt	output.txt
2 2 2 1 20 -3 2 2 0 1 11 1 20 -10	1 0
3 1 6 1 6 0 1 7 2 6 0 1 100 10 20 10	0.166666666666667 0.583333333333333 0.799600378342187

Задача 10. Антиплагиат

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

В специализированной художественной школе студентам в качестве выпускной работы необходимо нарисовать картину. За весь период обучения молодые художники освоили одну технику — рисунок мазками на квадратном холсте и два вида мазков: «звездочкой» и «решеткой». С помощью этой техники юным дарованиям необходимо изобразить квадратный шедевр $N \times N$, состоящий из таких мазков.

Задание весьма кропотливое, и каждый год находятся ушлые товарищи, норовящие воспользоваться тяжким трудом прошлых поколений. Полет фантазии у студентов ограниченный. Студент просто берёт чужую картину и несколько раз применяет следующие операции: 1) повернуть изображение на угол кратный 90 градусам, 2) отразить изображение относительно вертикальной или горизонтальной оси. То, что получилось, студент сдаёт как свою работу. Иногда студент вовсе может сдать работу прошлых лет как есть.

Профессора специализированной художественной школы чуют неладное, но, к сожалению, не в силах самостоятельно определить, являются ли две картины плагиатом или нет. Пора положить этому делу конец, написать программу, которая автоматизирует процесс, определяя, является ли одно изображение плагиатом другого или нет, и поможет профессорам вычислять нерадивых студентов.

Формат входных данных

В первой строке входного файла записано одно целое число: N — размер стороны квадратного холста ($1 \leqslant N \leqslant 500$).

В следующих N строках записано по N символов `*` либо `#`, означающих типы соответствующих мазков первой картины.

Далее идет одна пустая строка.

В следующих N строках описана вторая картина в аналогичном формате.

Формат выходных данных

В выходной файл необходимо вывести слово `YES`, если одна картина — плагиат другой, или `NO`, если нет.

Примеры

input.txt	output.txt
1 * # 	NO
3 #** #*# ### ### **# *##	YES