Внимание: Согласно правилам олимпиады, каждая команда может использовать только **один** компьютер. Вы можете использовать дополнительные компьютеры **только** для связи с сокомандниками и для чтения условий задач.

В любой момент времени может быть только один активный компьютер, на котором вы можете писать, компилировать, запускать программы. Активный компьютер может переключаться во время тура, например, чтобы следующую задачу писал другой член команды. Однако нельзя проявлять активность больше чем на одном компьютере одновременно.

Использовать интернет разрешается.

Использовать предварительно написанный код разрешается.

Внимание!

В этот тур включены задачи разного формата:

- Первые 6 задач имеют традиционный формат ICPC. Если решение полностью правильное, то задача засчитывается, и в рейтинг идёт одна зачётная единица.
- Последние 2 задачи оптимизационные. В каждой из них есть несколько уровней сложности, за преодоление каждого уровня даётся какое-то количество зачётных единиц.

Штрафное время в оптимизационной задаче начисляется за максимальный решённый уровень — по первой посылке, которая преодолела соответствующий барьер. Во время тура частота посылок по оптимизационным задачам может быть ограничена: например, не чаще раза в минуту.

Количество зачётных единиц по задачам:

1	2	3	4	5	6	7E	7M	7H	8E	8M	8H	8X
1	1	1	1	1	1	0.5	0.5	1	0.5	0.5	0.5	0.5

Задача 1. Драма

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

После прочтения античной драмы и произведений Шекспира Густав решил, что многие драматические произведения можно разделить на пять частей: экспозицию, развитие действия, кульминацию, развязку и постпозицию.

При таком делении *уровень конфликта* в произведении во время экспозиции постоянен, во время развития действия строго возрастает и достигает пика в кульминации, во время развязки строго убывает и, наконец, во время постпозиции остаётся постоянным.

Экспозиция и постпозиция могут происходить мгновенно, то есть занимать один момент времени. Развитие действия и развязка занимают минимум два момента времени, кульминация всегда одномоментна. Смотрите рисунки на следующей странице.

Для нескольких произведений Густав выписал уровни конфликта в хронологическом порядке и просит вас определить, соответствуют ли они уровням конфликта при вышеописанном делении. Если соответствуют, он просит вас сообщить моменты времени, когда

1. экспозиция переходит в развитие действия — t_1;
2. уровень конфликта достигает кульминации — t_2;
3. развязка переходит в постпозицию — t_3.

Формат входных данных

В первой строке входного файла записано одно целое положительное число D — количество произведений, проанализированных Густавом. Далее в отдельных строках следует описание этих произведений.

Первым в строке расположено целое число T_d — количество моментов времени, для которых выписаны уровни конфликта в произведении, за которым следует T_d целых положительных чисел c_t, соответствующих уровню конфликта в момент времени t ($3 \leqslant T_d \leqslant 10^5$, $1 \leqslant d \leqslant D, 1 \leqslant c_t \leqslant 2 \cdot 10^9$, $1 \leqslant t \leqslant T_d$).

Гарантируется, что сумма всех T_d не превосходит 10^5.

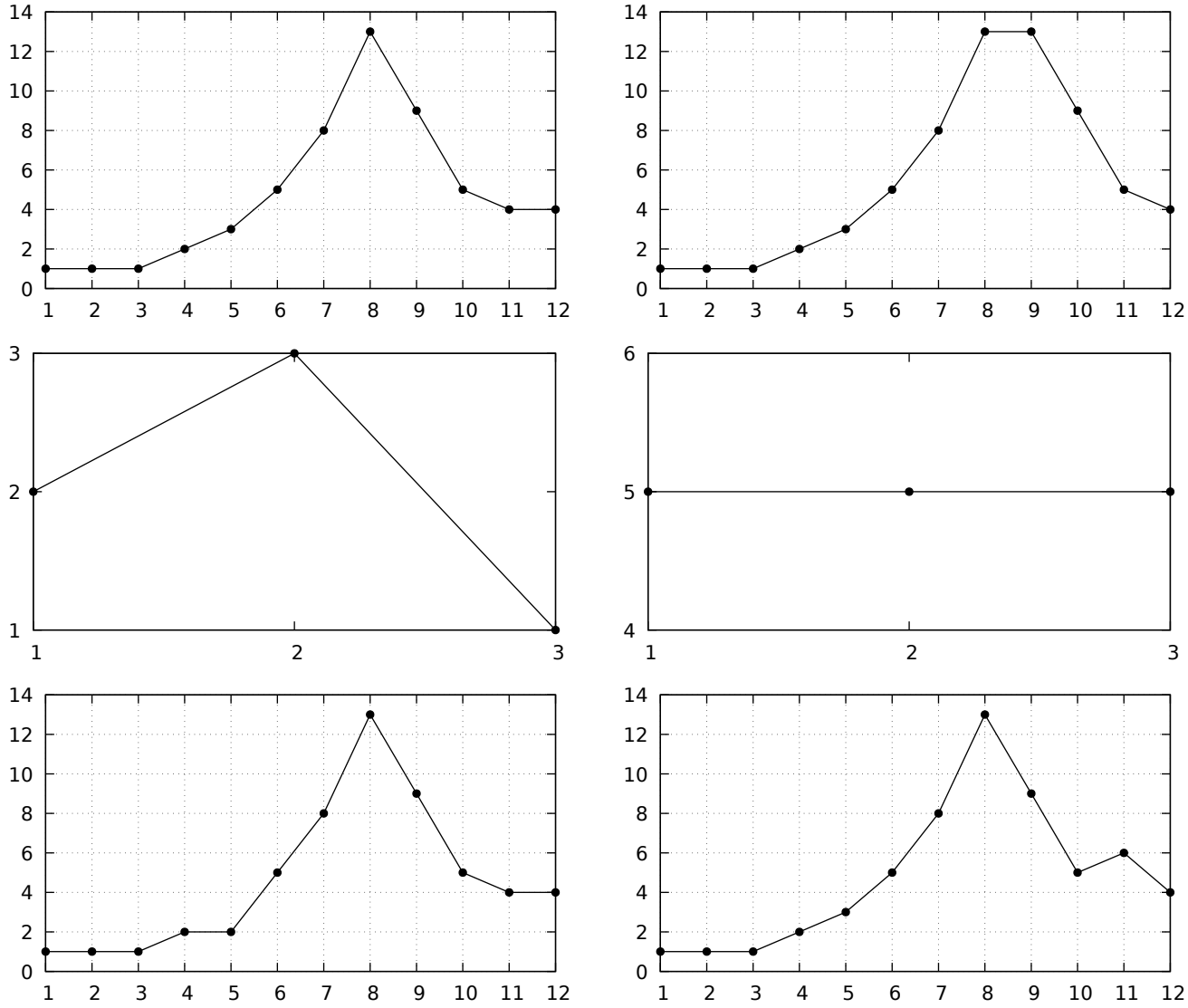
Формат выходных данных

Для каждого произведения в порядке следования во входном файле выведите в отдельной строке выходного файла: `Freytag`, если уровни конфликта в произведении изменяются в соответствии с делением Густава, и `Nein` иначе. В случае ответа `Freytag` дополнительно выведите три целых положительных числа t_1, t_2, t_3 — вышеописанные моменты времени ($1 \leqslant t_i \leqslant T_d$, $1 \leqslant i \leqslant 3$, $t_1 < t_2 < t_3$).

Примеры

input.txt	output.txt
6 12 1 1 1 2 3 5 8 13 9 5 4 4 12 1 1 1 2 3 5 8 13 13 9 5 4 3 2 3 1 3 5 5 5 12 1 1 1 2 2 5 8 13 9 5 4 4 12 1 1 1 2 3 5 8 13 9 5 6 4	Freytag 3 8 11 Nein Freytag 1 2 3 Nein Nein Nein

Пояснение к примеру



Графики в порядке сверху вниз, слева направо соответствуют произведениям из примера входных данных. По горизонтальной оси отложены моменты времени, по вертикальной — уровни конфликта в произведениях.

1. Отрезок времени $[1;3]$ — экспозиция, $[3;8]$ — развитие действия, в момент времени 8 происходит кульминация, $[8;11]$ — развязка, $[11;12]$ — постпозиция.
2. После развития действия и кульминации не наступает сразу развязка.
3. Отрезок времени $[1;1]$ — экспозиция, $[1;2]$ — развитие действия, в момент времени 2 происходит кульминация, $[2;3]$ — развязка, $[3;3]$ — постпозиция.
4. Отсутствуют развитие действия, кульминация и развязка.
5. В зависимости от интерпретации либо при развитии действия нет строгого возрастания уровня конфликта на отрезке $[4;5]$, либо после кульминации в момент времени 4 не начинается развязка.
6. В зависимости от интерпретации либо после момента времени 10 в постпозиции не сохраняется постоянный уровень конфликта, либо в развязке уровень конфликта не убывает на отрезке $[10;11]$.

Задача 2. Сумма произведений

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

Назовём *устремлённостью* последовательности чисел произведение минимального её элемента на максимальный её элемент. Назовём *подотрезком* последовательности любой набор подряд идущих её элементов.

Дана последовательность и число M. Требуется рассмотреть все её подотрезки, на которых сумма чисел больше или равна M, посчитать для каждого из них устремлённость, и вывести сумму всех этих устремлённостей.

Формат входных данных

В первой строке входного файла записано два целых числа: N — количество элементов в заданной последовательности и M — ограничение снизу на сумму на подотрезке ($1 \leqslant N \leqslant 10^5$, $1 \leqslant M < 10^{14}$).

Во второй строке записана последовательность из N целых чисел. Каждое число положительное и строго меньше 10^9.

Формат выходных данных

В выходной файл необходимо вывести одно целое число — сумму устремлённостей всех подходящих подотрезков.

Примеры

input.txt	output.txt
5 10 2 5 3 7 5	157
5 1 2 5 3 7 5	294

Пояснение к примеру

При $N = 5$ возможно 15 подотрезков. Ниже приведены их устремлённости в формате минимум · максимум. В каждой строке указаны подотрезки фиксированной длины L в порядке их следования во входной последовательности слева направо.

$L = 1$:	$2 \cdot 2$		$5 \cdot 5$		$3 \cdot 3$		$7 \cdot 7$		$5 \cdot 5$
$L = 2$:		$2 \cdot 5$		$3 \cdot 5$		$\underline{3 \cdot 7}$		$\underline{5 \cdot 7}$	
$L = 3$:			$\underline{2 \cdot 5}$		$\underline{3 \cdot 7}$		$\underline{3 \cdot 7}$		
$L = 4$:				$\underline{2 \cdot 7}$		$\underline{3 \cdot 7}$			
$L = 5$:					$\underline{2 \cdot 7}$				

В первом тесте рассматриваются только подотрезки с суммой 10 и больше, на картинке их устремлённости подчёркнуты. Во втором тесте ограничение снизу равно 1, значит, все подотрезки подходят.

Задача 3. Остап и политические игры

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

После сочинения шарад и ребусов Синицкий занялся более серьёзными вещами — придумал занимательную настольную игру. В этой игре два игрока двигают одну фишку по прямоугольному клеточному полю размера M на N, заполненному числами.

Игроки ходят по очереди. За каждый ход игрок должен передвинуть фишку либо на клетку вниз, либо на клетку вправо. Изначально фишка расположена в левом верхнем углу поля. Игра заканчивается, когда фишка оказывается в правом нижнем углу поля. Двигать фишку за пределы поля нельзя.

При этом, поскольку первый игрок олицетворяет собой пролетариат, а второй — буржуазию, то и цели игры у них диаметрально противоположные. Первый хочет максимизировать сумму чисел на пройденном пути, второй игрок хочет эту же сумму минимизировать. Этому Синицкого научила политически подкованная внучка Зося. Сам он эту игру только придумал, а играть так и не научился.

Остап хочет непременно освоить эту новую игру, чтобы завоевать расположение Зоси, а также реабилитироваться после неудачного сеанса одновременной игры в шахматы.

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.

Сначала вам необходимо прочитать из стандартного потока ввода информацию об игровом поле.

В первой строке записано два целых числа M и N — количество строк и столбцов на игровом поле соответственно ($2 \leqslant M, N \leqslant 200$).

Далее идут M строк по N целых чисел в каждой — числа, записанные в клетках игрового поля. Все числа не превышают 10^6 по абсолютной величине.

После этого начинается игра. Вы играете за первого игрока и стремитесь максимизировать сумму очков. Интерактор играет за второго игрока и стремится минимизировать сумму, хотя при этом может делать неоптимальные для себя ходы. Всего должно быть совершено $M + N - 2$ ходов.

В свой ход необходимо выводить целое число s и букву t. Число s должно быть равно максимальной сумме очков, которую можно получить в конце игры, если начиная с этого момента оба игрока будут играть оптимально. Буква t определяет, куда ходить, в формате: `R` (вправо) или `D` (вниз). Выведенный вами ход должен быть оптимальным, то есть после него должно быть по-прежнему возможно набрать s очков при оптимальной игре обоих игроков.

В свой ход программа-интерактор выводит одну букву `R` или `D`, обозначающую, куда второй игрок двигает фишку.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного хода. 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
3 4 5 8 5 4 1 4 1 2 6 3 7 3 - D - R -	- - - - 27 R - 30 D - 30 R

Пояснение к примеру

В игре совершается 5 ходов, из которых первый игрок делает 3. На втором ходу второй игрок совершает неоптимальный ход, двигая фишку вниз, из-за чего конечный результат увеличивается на 3. На четвёртом ходу он двигает фишку вправо.

Задача 4. Остап и P=NP

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

После фиаско со стульями и Корейко Остап решил заработать миллион гораздо более простым путем. Всем известно, что за решение задачи P=NP полагается миллион долларов. И он решил взяться за эту задачу. Для начала решил вникнуть, что это за классы такие: P и NP. Остапу пришлось просветиться про показательные функции, степенные и что показательная функция растет быстрее степенной.

А вот с этого момента у него возникло непонимание. Разумеется, что при x, больших некоторого x_0, a^x будет больше x^b. Что же будет при других x? Это тот ещё вопрос!

Вот великого комбинатора и заинтересовало: сколько рациональных x удовлетворяют неравенству $x^b > a^x$. Остап знает только обычные несократимые дроби вида p/q, да и то, если у них небольшой знаменатель $q \leqslant Q$.

Формат входных данных

В первой строке входного файла записано два вещественных числа a и b — основание показательной функции и степень степенной ($2 \leqslant a, b \leqslant 100$, $a \neq b$). У обоих чисел не больше двух знаков после десятичной точки.

Во второй строке записано одно целое число Q — ограничение на знаменатель дробей ($1 \leqslant Q \leqslant 10\,000$).

Гарантируется, что если для дроби $x = p/q$ со знаменателем $q \leqslant Q$ выполняется неравенство $x^b > a^x$, то и для числа $x \pm \delta$ оно также выполняется для любого $|\delta| \leqslant 10^{-9}$. Аналогично, если неравенство не выполняется для дроби $x = p/q$, то для близких чисел $x \pm \delta$ оно тоже не выполняется при тех же δ.

Формат выходных данных

В выходной файл должно быть записано одно целое число — ответ на задачу, а именно, количество положительных рациональных чисел $x = p/q$, для которых выполнены неравенства $x^b > a^x$ и $q \leqslant Q$.

Примеры

input.txt	output.txt
2 3 1	8
3 4.01 2	11

Задача 5. Порталы видимости

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

В игровом движке idTech4 геометрия уровней строится из твердотельных выпуклых многогранников. Пространство, не заполненное этими многогранниками, создатель уровня разделяет на отдельные *игровые области* с помощью *порталов видимости*. К сожалению, авторы уровней часто ошибаются при расстановке порталов, в результате чего проблемные порталы просто выбрасываются. Найти эту ошибку без дополнительной диагностики бывает весьма трудно. В данной задаче вам предлагается реализовать диагностику в упрощённом двумерном случае.

Будем считать, что уровень представляется клеточным полем на плоскости. Все клетки уровня пустые. Между любыми двумя соседними клетками может стоять непроходимая стена, портал видимости, или может быть пусто. Если между соседними клетками пусто, то разрешается переместиться из одной в другую за один шаг, а если что-то стоит, то перемещаться нельзя. Согласно описанному правилу перемещения, все клетки уровня разбиваются на компоненты связности — это и есть игровые области.

Две клетки по разные стороны от правильно расположенного портала должны попадать в разные игровые области. Если они попадают в одну игровую область, то рассматриваемый портал расположен неверно. Чтобы создатель уровня мог найти и исправить эту проблему, следует вывести для него *диагностический путь* — путь, который соединяет две этих клетки. Разумеется, этот путь не должен пересекать стены и порталы.

Дана карта уровня. Требуется определить, какие порталы расположены неверно, и найти для каждого их них диагностический путь с минимальным количеством перемещений.

Формат входных данных

В первой строке входного файла записано три целых числа: M, N — количество строк и столбцов соответственно на карте уровня, и K — количество порталов ($1 \leqslant M, N \leqslant 3\,000$, $1 \leqslant K \leqslant 500\,000$).

Далее приведена карта уровня, которая описана в виде $(2M+1)$ строк по $(2N+1)$ символов в каждой. Используются следующие символы:

- ‘`*`’ (ASCII 42) — вершина между четырьмя соседними клетками,
- ‘`.`’ (ASCII 46) — пустая клетка или пустота между соседними клетками,
- ‘`-`’ (ASCII 45) — стенка между соседними клетками из разных строк,
- ‘`|`’ (ASCII 124) — стенка между соседними клетками из разных столбцов,
- ‘`~`’ (ASCII 126) — портал между соседними клетками из разных строк,
- ‘`$`’ (ASCII 36) — портал между соседними клетками из разных строк.

У каждого символа карты можно определить номер строки r и номер столбца c (нумерация начинается с единицы). Когда оба номера чётные, то стоит символ точки и обозначает пустую клетку. Когда оба номера нечётные, то стоит символ звёздочки и ничего не обозначает. Когда чётность номеров различается, символ обозначает наличие стенки или портала между соседними пустыми клетками. Если между клетками пусто, то стоит точка, если находится стена, то стоит минус или вертикальная черта, а если портал, то стоит тильда или доллар.

Гарантируется, что на карте ровно K порталов. Никакой портал не находится на границе уровня.

Формат выходных данных

В каждую из K строк выходного файла требуется вывести искомый диагностический путь. Пути для порталов требуется выводить в том порядке, в котором символы, обозначающие порталы, встречаются во входных данных. То есть порталы упорядочиваются сначала по номеру строки на карте уровня, а в случае равенства — по номеру столбца. Если портал расположен верно и диагностического пути для него нет, тогда вместо пути следует вывести слово `OK`.

Диагностический путь может начинаться с любой из двух клеток, между которыми стоит портал. Путь, проходящий через $T + 1$ клетку, выводится в виде строки длины T, в которой каждый символ обозначает одно перемещение:

- '`d`' — увеличить номер строки,
- '`u`' — уменьшить номер строки,
- '`r`' — увеличить номер столбца,
- '`l`' — уменьшить номер столбца.

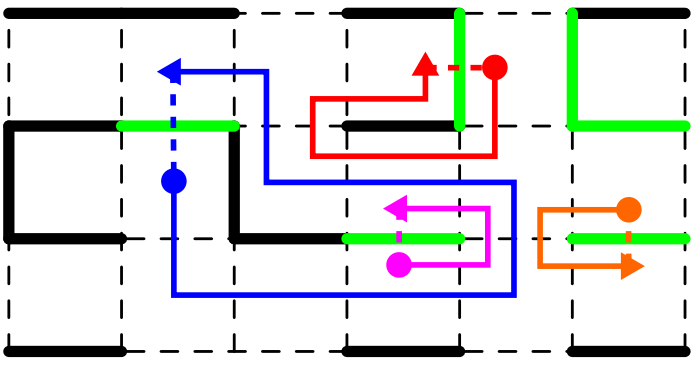
Гарантируется, что в правильном ответе сумма квадратов T по всем диагностическим путям не превышает 10^7.

Примеры

input.txt	output.txt
3 6 6 *-*-*.*-*.*-* ....$.$.. *-*~*.*-*.*~* |...|..... *-*.*-*~*.*~* ..... *-*.*.*-*.*-*	dllur OK drrrullul OK rul ldr

Иллюстрация

На картинке показан пример: стенки чёрные, порталы зелёные. Для некорректных порталов показаны диагностические пути разными цветами.



Задача 6. Остап и Стоунхендж

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

После тостов за ирригацию и мелиорацию Узбекистана гости разошлись, и Остап вместе с любимой астролябией пошел чего-нибудь да померить. В тоске он добрел до развалин райсобеса, им же и разваленного, где и обнаружил большое количество камней, беспорядочно раскиданных по ровному полю. И над всей этой печальной картиной величественно восходила луна. Тут сын турецо-подданного вспомнил про Стоунхендж и подумал: «А мы-то чем хуже? У нас вона чего есть!!!». И принялся лихорадочно искать пару камней, направление между которыми лучше всего бы указывало на направление восхода луны. Благо и астролябия под боком. Только задача эта непростая, с кондачка ее не решить. Тут нужен строго научный подход, а академиев Остап, увы, не кончал. Надо бы ему помочь.

Формат входных данных

В первой строке входного файла записано три целых числа: N — количество камней, a и b — компоненты вектора, задающего направление восхода луны ($2 \leqslant N \leqslant 10^5$, $-10^5 \leqslant a, b \leqslant 10^5$, $a^2 + b^2 > 0$).

В оставшихся N строках описаны камни, по одному камню в строке. Для i-ого камня указано два целых числа x_i и y_i — его координаты ($-10^5 \leqslant x_i, y_i \leqslant 10^5$).

Гарантируется, что нет двух камней с одинаковыми координатами.

Формат выходных данных

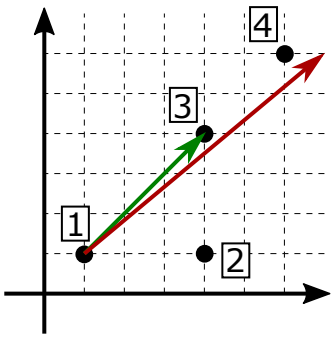
В выходной файл следует вывести два различных целых числа — номера камней i, j, таких, что угол между вектором (a, b) и вектором $(x_j - x_i, y_j - y_i)$ минимально возможный. Также разрешается вывести пару камней, для которых угол отличается от оптимального не более чем на 10^{-10} радиан.

Камни пронумерованы от 1 до N в порядке задания во входном файле.

Пример

input.txt	output.txt
4 6 5 1 1 4 1 4 4 6 6	1 3

Пояснение к примеру



На картинке изображены: камни с номерами, вектор, идущий от камня 1 до камня 3 и вектор направления восхода луны (6, 5).

Ответы **1 4** и **3 4** тоже правильные, а ответ **3 1** — нет, так как в последнем случае угол будет тупым.

Задача 7. Средний цвет

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

Внимание: это оптимизационная задача без идеального решения, и она влияет на рейтинг особым образом.

Имеется неизвестное изображение размером H пикселей в высоту и W пикселей в ширину. Рассмотрим пиксель в строке i и столбце j: его значение задаётся тремя числами R_{ij}, G_{ij} и B_{ij} — значениями красного, зелёного и синего соответственно.

Ваша задача — найти средний цвет изображения как можно более точно. Средний цвет — это тройка чисел R_*, G_*, B_*:

$$R_* = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} R_{ij}$$

$$G_* = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} G_{ij}$$

$$B_* = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} B_{ij}$$

Из всех HW пикселей изображения вам изначально известно K пикселей, и ещё Q пикселей по своему выбору вы можете узнать дополнительно. Про позиции известных пикселей в каждом тесте гарантируется:

- они одинаковы для всех участников и для всех посылок;
- они выбраны случайным образом согласно некоторому распределению;
- при генерации не использовалась информация о цветах пикселей.

Примеры тестов можно увидеть в архиве с примерами.

Ваша программа должна выдать в конце работы оценку среднего цвета изображения: R_{avg}, G_{avg}, B_{avg}.

Ошибка на этом тесте вычисляется как квадрат расстояния между правильным ответом и вашей оценкой: $(R_{avg} - R_*)^2 + (G_{avg} - G_*)^2 + (B_{avg} - B_*)^2$.

Суммарная ошибка решения — это сумма ошибок по всем тестам. Всего в тестирующей системе 27 тестов.

В задаче есть три уровня сложности, и чтобы решить каждый уровень, нужно сдать решение с суммарной ошибкой меньше заданного барьера:

- Easy: 200.0
- Medium: 50.0
- Hard: 20.0

При этом решение должно корректно завершиться на всех тестах. За преодоление каждого барьера вы получаете зачётные единицы в рейтинге: $+\frac{1}{2}$ за Easy и Medium, и $+1$ за Hard. Всего можно заработать две зачётные единицы.

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.

Сначала ваша программа должна прочитать входные данные.

В первой строке записано четыре целых числа: H — высота изображения, W — ширина изображения, K — количество изначально известных пикселей, Q — сколько пикселей можно узнать дополнительно. Эти числа во всех тестах одинаковы: $H = W = 512$, $K = 2000$, $Q = 2000$.

В следующих K строках описаны изначально известные пиксели, по одному в строке.

В каждой из этих строк записано пять целых чисел: i, j — номер строки и столбца соответственно, и r, g, b — цвет пикселя как значения красного, зелёного и синего соответственно ($1 \leqslant i \leqslant H$, $1 \leqslant j \leqslant W$, $0 \leqslant r, g, b \leqslant 255$).

Гарантируется, что все известные пиксели различны.

Далее ваша программа должна сделать не более Q запросов о цветах пикселей.

Каждый запрос описывается тремя целыми числами: 0 i j, где i и j — номер строки и столбца соответственно ($1 \leqslant i \leqslant H$, $1 \leqslant j \leqslant W$).

В ответ интерактор выводит три целых числа r, g, b — цвет заданного пикселя как значения красного, зелёного и синего ($0 \leqslant r, g, b \leqslant 255$).

В конце работы следует вывести ответ в формате: 1 R_{avg} G_{avg} B_{avg}, где вещественные числа R_{avg} G_{avg} B_{avg} определяют вашу оценку среднего цвета.

Нарушение протокола интеракции со стороны вашей программы влечёт вердикт Wrong Answer. Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного хода. Иначе решение может получить вердикт Timeout.

Примеры

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

стандартный поток ввода	стандартный поток вывода
3 3 2 1 1 1 255 255 255 1 3 0 0 0 - 10 20 30 -	- - - 0 2 2 - 1 88.3333333 91.6666666 95

Пояснение к примеру

Этого примера в тестах нет. В нём для простоты выбраны значения H, W, K и Q, отличные от тех, что будут в тестах.

Задача 8. Сжатие изображений

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

Внимание: это оптимизационная задача без идеального решения, и она влияет на рейтинг особым образом.

Анонсирован инновационный формат TruPEG5 для сжатия изображений с потерей данных. Вам предлагается создать программу для сжатия в этот формат.

Изображение представляет матрицу пикселей размера H на W, цвет каждого пикселя записан в RGB8 формате, то есть в каждом пикселе хранится три целочисленных значения в диапазоне от 0 до 255 включительно: красная, зелёная и синяя компоненты цвета соответственно. В формате TruPEG5 изображение делится на квадратные блоки размера 5 на 5 пикселей, и каждый блок кодируется независимо от остальных.

В сжатом виде блок представляется следующими данными:

1. Опорный цвет A, представленный в RGB8-формате с компонентами A_r, A_g и A_b.
2. Опорный цвет B, представленный в RGB8-формате с компонентами B_r, B_g и B_b.
3. 25 индексов в диапазоне от 0 до 4 включительно, по одному индексу для каждого пикселя блока.

Отображаемый цвет C пикселя в сжатом блоке определяется по формуле:

$$\begin{cases} C_r &= \left\lfloor A_r + \frac{k}{4}\Big(B_r - A_r\Big) \right\rfloor \\ C_g &= \left\lfloor A_g + \frac{k}{4}\Big(B_g - A_g\Big) \right\rfloor \\ C_b &= \left\lfloor A_b + \frac{k}{4}\Big(B_b - A_b\Big) \right\rfloor \end{cases}$$

Здесь k — это индекс рассматриваемого пикселя, а C_r, C_g, C_b — компоненты цвета C в RGB8-формате. Скобки $\lfloor\cdot\rfloor$ обозначают округление вниз до ближайшего целого.

Погрешность сжатия определяется следующим образом. Если исходный пиксель имеет цвет P, а его отображаемый цвет в сжатом изображении равен C, тогда погрешность этого пикселя равна евклидову расстоянию:

$$Err = \sqrt{(P_r - C_r)^2 + (P_g - C_g)^2 + (P_b - C_b)^2}$$

Погрешность всего изображения равна:

$$TotalErr = \sqrt{\frac{\sum\limits_{i,j} Err^2}{HW}}$$

То есть вычисляется сумма квадратов погрешностей всех пикселей, делится на количество пикселей, и из отношения берётся квадратный корень.

Чем меньше погрешность сжатого изображения, тем лучше. Баллы за тест определяются как отношение погрешности решения жюри и вашего решения. Таким образом, если решение выдаёт в три раза большую погрешность, чем решение жюри, то оно получает одну треть балла за тест.

Баллы за всю задачу определяются как сумма баллов решения за все тесты. В задаче есть четыре уровня сложности, и чтобы решить каждый уровень, нужно набрать суммарные баллы больше заданного барьера:

- Easy: 20
- Medium: 27.5
- Hard: 29.6
- Extreme: 30.1

При этом решение должно корректно завершиться на всех тестах. За преодоление каждого барьера вы получаете $+\frac{1}{2}$ зачётных единицы в рейтинге. Всего можно заработать две зачётных единицы.

Формат входных данных

В первой строке входного файла записано два целых числа H и W — высота и ширина изображения соответственно. Остальные H строк описывают изображение, в каждой по $3W$ целых чисел в диапазоне от 0 до 255 включительно. Первые три числа определяют RGB-компоненты первого пикселя в строке, следующие три числа — второго пикселя, и так далее.

Гарантируется, что размеры H и W делятся нацело на 5 и не превышают 500.

Тесты по задаче делятся на три группы, в каждой группе по 10 тестов схожего характера. Вы можете увидеть первый тест из каждой группы в архиве с примерами. Архив также содержит скрипты на Python для конвертации:

1. из формата входных данных в BMP формат,
2. из BMP формата в формат входных данных,
3. из сжатого формата выходных данных в несжатый формат входных данных.

Погрешность решения жюри на тестах:

1	2	3	4	5	6	7	8	9	10
9.20331	6.66626	6.36300	4.15339	8.09847	7.17434	5.68570	6.59537	4.80343	8.03184
11	12	13	14	15	16	17	18	19	20
0.79054	0.94050	0.98891	0.94506	0.83096	0.60333	1.08246	1.0964	0.78091	1.00389
21	22	23	24	25	26	27	28	29	30
12.78183	30.67424	31.87592	16.07557	16.47340	14.27318	0.19252	0.26503	0.17634	0.21843

Формат выходных данных

В первой строке выходного файла должно быть записано два целых числа H и W — высота и ширина изображения. Остальные $HW/25$ строк описывают сжатые блоки. Сначала следует вывести все $W/5$ блоков, покрывающие первые (верхние) пять строк изображения, в порядке слева направо. Затем $W/5$ блоков, покрывающих строки с шестой по десятую, и так далее. Направления лево, право и верх определяются так, как их видно в текстовом файле.

Для каждого блока выведите 31 целое число. Первые шесть чисел A_r, A_g, A_b, B_r, B_g, B_b задают опорные цвета блока и должны быть в диапазоне от 0 до 255 включительно. Остальные 25 чисел задают индексы пикселей блока и должны быть в диапазоне от 0 до 4 включительно. Индексы пикселей следует выводить в следующем порядке: сначала индексы пикселей самой верхней строки слева направо, затем индексы второй сверху строки слева направо, и так далее.

Пример

input.txt

```
10 5
254 253 252  191 191 191  191 191 191  191 191 191  191 191 191
191 191 191  191 191 191  191 191 191  127 127 127  127 127 127
127 127 127  127 127 127  127 127 127  63  63  63   63  63  63
63  63  63   63  63  63   63  63  63   63  63  63   63  63  63
0   0   0    0   0   0    0   0   0    0   0   0    0   0   0
100 120 140  96  113 120  92  106 101  88  99  82   85  93  63
96  113 120  92  106 101  88  99  82   85  93  63   85  93  63
92  106 101  88  99  82   85  93  63   85  93  63   85  93  63
88  99  82   85  93  63   85  93  63   85  93  63   85  93  63
85  93  63   85  93  63   85  93  63   85  93  63   80  82  77
```

output.txt

```
10 5
255 255 255  0  0  0  0 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4
100 120 140 85 93 63  0 1 2 3 4 1 2 3 4 4 2 3 4 4 4 3 4 4 4 4 4 4 4 4
```

Пояснение к примеру

Пример выше приведён лишь для пояснения, при проверке решения в системе его нет.

Ниже показано, как отображается сжатое изображения из выходных данных. Заметим, что от входного изображения отличаются только два пикселя: в верхнем левом и нижнем правом углах. Первый из них отличается на (1, 2, 3), что даёт погрешность $\sqrt{1+4+9} \approx 3.741657$. Второй отличается на (-5, -11, 14), что даёт погрешность $\sqrt{342} \approx 18.493242$. Всего пикселей 50, значит, погрешность всего изображения вычисляется как $\sqrt{(3.741657^2 + 18.493242^2)/50} \approx 2.668332$.

```
255 255 255  191 191 191  191 191 191  191 191 191  191 191 191
191 191 191  191 191 191  191 191 191  127 127 127  127 127 127
127 127 127  127 127 127  127 127 127  63  63  63   63  63  63
63  63  63   63  63  63   63  63  63   63  63  63   63  63  63
0   0   0    0   0   0    0   0   0    0   0   0    0   0   0
100 120 140  96  113 120  92  106 101  88  99  82   85  93  63
96  113 120  92  106 101  88  99  82   85  93  63   85  93  63
92  106 101  88  99  82   85  93  63   85  93  63   85  93  63
88  99  82   85  93  63   85  93  63   85  93  63   85  93  63
85  93  63   85  93  63   85  93  63   85  93  63   85  93  63
```