Problem 1. Сложная документация

Input file:	input.txt
Output file:	output.txt
Time limit:	1 секунда
Memory limit:	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.

Input

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

Output

Выведите в выходной файл исправленный текст.

Examples

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 :

Commentary

Ни одна строка в примере не заканчивается пробелом.

Problem 3. Найти радистку

Input file:	`стандартный поток ввода`
Output file:	`стандартный поток вывода`
Time limit:	1 секунда
Memory limit:	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 $\varphi \geq 90$ 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.

Interaction Protocol

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 ($|x_0|, |y_0|, |x_d|, |y_d| \leq 10^4$). 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.

Example

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

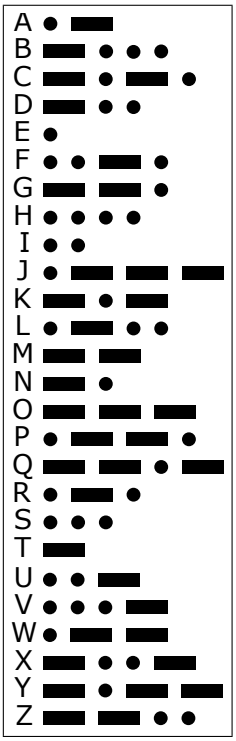
стандартный поток ввода	стандартный поток вывода
- 0.035999999999999972799536 - 0.000000000000000000000000 - 0.008999999999999993199884 - 0.089999999999999827915431 - 0.166666666666666574148081 - -1	? 0 0 0 1 - ? -2 -3 -1 1 - ? -4 -3 0 100 - ? 3 0 3 1 - ? 4 0 0 1 - ? 4 3 1 1

Example explanation

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.

Problem 4. Код Морзе

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 секунды
Memory limit:	256 мегабайт



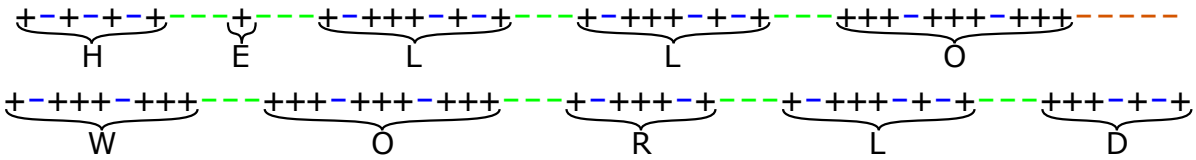
В азбуке Морзе каждая буква представляется последовательностью «точек» и «тире». Однако одного лишь этого недостаточно, чтобы передавать текст по радио. Помимо этого, код Морзе предписывает длительности для точки и тире, а также для различных пауз между ними.

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.

Input

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

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.

Output

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 $K \geq 0$ — 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.

Examples

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

Example explanation

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.

Problem 5. Леспромхоз

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 секунды
Memory limit:	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.

Input

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

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.

Output

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 ($1 \le k_i \le N$, $0 \le \varphi_i \le 360$).

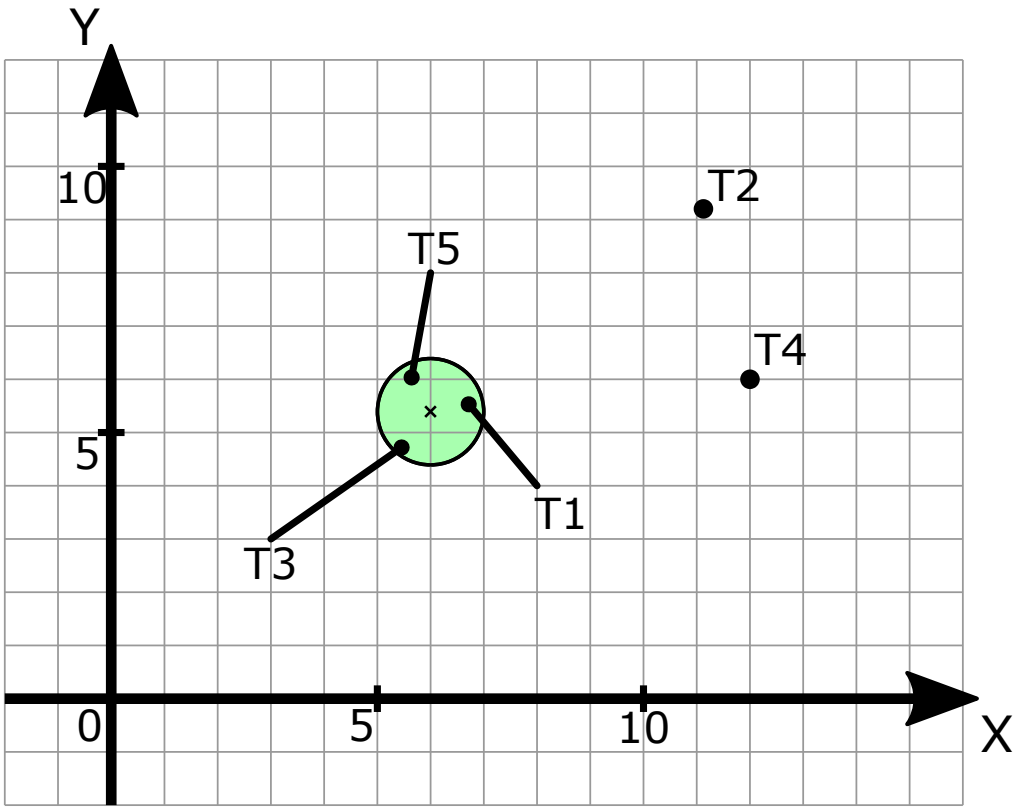
Деревья нумеруются в порядке описания во входном файле, начиная с единицы. Порядок вывода деревьев значения не имеет. Полярный угол измеряется начиная от оси X в направлении оси Y, выводится в градусах, и определяет направление от начала сваленного дерева к его концу.

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

Examples

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

Illustration



Problem 7. Планировщик

Input file:	input.txt
Output file:	output.txt
Time limit:	2 секунды
Memory limit:	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.

Input

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

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

Output

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

Examples

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

Problem 9. Решающий удар

Input file:	input.txt
Output file:	output.txt
Time limit:	3 секунды
Memory limit:	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.

Input

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

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

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 ($1 \le n \le 10$, $2 \le f \le 20$, $-10 \le m \le 10$).

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

Output

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}.

Examples

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

Problem 10. Антиплагиат

Input file:	input.txt
Output file:	output.txt
Time limit:	1 секунда
Memory limit:	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.

Input

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

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.

Output

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

Examples

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

Problem 11. Список файлов

Input file:	input.txt
Output file:	output.txt
Time limit:	1 секунда
Memory limit:	256 мегабайт

Задан набор файлов, необходимо вывести количество файлов для каждого из расширений. Расширением файла называется последовательность символов в имени файла после символа точки.

Файловая система чувствительна к регистру. Даже если имена файлов отличаются только регистром символов, они всё равно считаются различными.

Input

В первой строке входного файла дано целое число N "— количество имён файлов ($1 \le N \le 10^3$).

Далее в каждой из следующих N строк находится имя файла длиной не больше 200 символов. Имя файла состоит только из латинских символов верхнего и нижнего регистров, цифр и символа точки '.'. Гарантируется, что символ точки встречается в имени файла ровно один раз. Также до и после символа точки в имени файла присутствует не менее одного символа. Гарантируется, что во входном файле каждое имя встречается ровно один раз.

Output

Для каждого из расширений, присутствующих во входном файле, вывести в выходной файл количество файлов с таким расширением в виде: <расширение>:␣<количество>. Расширения требуется выводить в порядке первого упоминания во входном файле.

Example

input.txt	output.txt
6 218052.pdf Movie00.mkv Invoice.xls book.pdf book.epub Movie01.mkv	pdf: 2 mkv: 2 xls: 1 epub: 1

Example explanation

Два файла с расширением pdf: 218052.pdf, book.pdf.

Два файла с расширением mkv: Movie00.mkv, Movie01.mkv.

Один файл с расширением xls: Invoice.xls.

Один файл с расширением epub: book.epub.

Problem 12. Улей

Input file:	input.txt
Output file:	output.txt
Time limit:	1 секунда
Memory limit:	256 мегабайт

На плоскости, замощённой правильными шестиугольниками, заданы соты пчелиного улья. Рабочие пчёлы сначала неверно поняли проектную документацию, и теперь им необходимо повернуть улей вокруг соты с пчелиной королевой на 60 градусов по часовой стрелке.

Улей состоит из гексагональных сот. Все соты ориентированы так, что сверху и снизу расположены вершины шестиугольника, а слева и справа рёбра, которыми сота соединена с соседними сотами в ряду. Каждый следующий ряд смещён относительно предыдущего на половину соты. Ось Ox направлена вдоль горизонтального ряда сот слева направо. Ось Oy направлена вверх под углом 60 градусов к оси Ox. Оси пересекаются в соте с координатами $(0, 0)$. Также в пояснении к примеру приложены иллюстрации, на которых изображена нумерация ячеек.

Input

В первой строке входного файла записано три целых числа N, X и Y, где N "— количество сот в улье (кроме соты королевы), X и Y "— координаты соты королевы ($1 \le N \le 10^4; |X|, |Y| \le 10^4$). Далее в каждой из следующих N строк находятся пары целых чисел x и y "— координаты очередной соты пчелиного улья ($|x|, |y| \le 10^4$). Координаты всех сот во входном файле различны.

Output

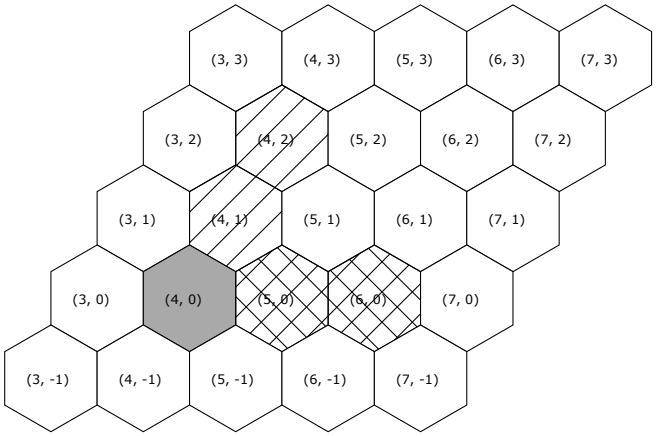
Для каждой из N сот в порядке перечисления во входном файле необходимо вывести в выходной файл на отдельной строке два целых числа "— её координаты после поворота.

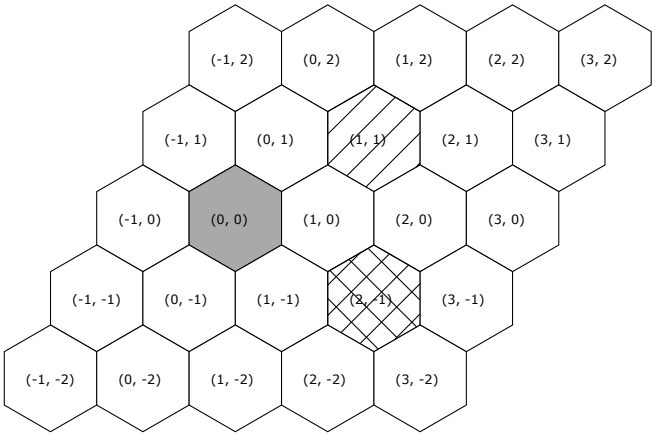
Example

input.txt	output.txt
2 4 0 4 1 4 2	5 0 6 0
1 0 0 1 1	2 -1

Example explanation

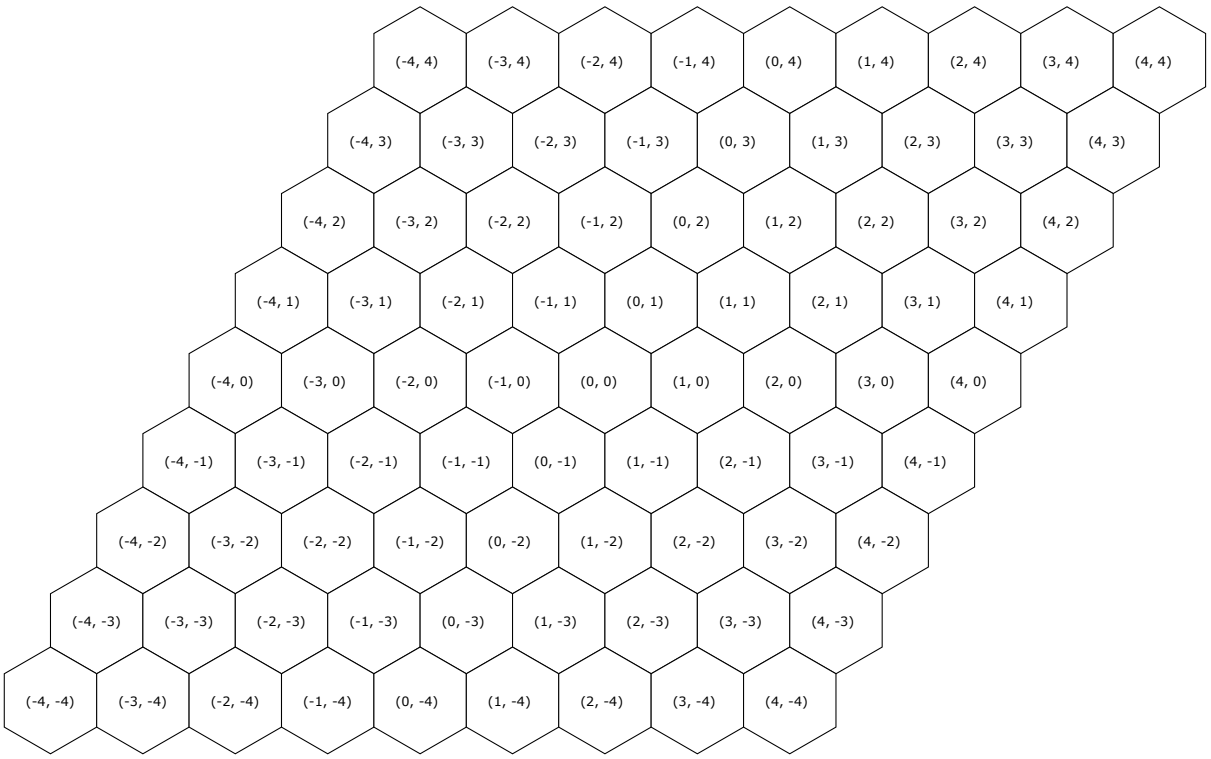
Ульи из примеров представлены на рисунках ниже. Сота пчелиной королевы обозначена серым цветом, соты в начальном положении заштрихованы параллельными прямыми, соты после поворота заштрихованы пересекающимися прямыми.





Illustration

На этом улье можно рисовать. ;-)



Problem 13. Побочные эффекты

Input file:	input.txt
Output file:	output.txt
Time limit:	1 секунда
Memory limit:	256 мегабайт

Программист разрабатывает приложение. Поскольку он состоит в клубе «Юные друзья функционального программирования», то хочет знать, сколько из его функций содержат побочные эффекты.

В начальный момент времени про функции из программы известно, присутствуют в них побочные эффекты или нет. Также в начальный момент времени ни одна функция не вызывает другую. Но программист решает изменить логику работы приложения и начинает вставлять вызовы одних функций в другие. Новые функции программист не пишет. Если функция вызывает функцию с побочными эффектами, то вызывающая функция также начинает обладать побочными эффектами, и так далее по цепочке вызовов. Рекурсия в вызовах допустима. Также из одной функции можно вызывать другую несколько раз.

Необходимо определить, сколько функций с побочными эффектами присутствует в программе после каждого добавления вызова функции программистом.

Input

В первой строке входного файла записано три целых числа N, K и M, где N "— общее количество функций, K "— начальное количество функций с побочными эффектами, M "— количество вызовов функций, добавляемых программистом ($1 \le N, K, M \le 10^5; K \le N$). Функции нумеруются по порядку от 1 до N.

Далее в одной строке следуют K различных чисел от 1 до N "— номера функций с побочными эффектами в начальный момент времени.

В каждой из следующих M строк находятся пары целых чисел a и b, которые означают, что программист добавил в программу вызов функции с номером b из функции с номером a ($1 \le a, b \le N$).

Output

Для каждого из M добавлений вызова функции в выходной файл необходимо вывести на отдельной строке количество функций с побочными эффектами на момент после добавления этого вызова.

Example

input.txt	output.txt
3 1 5 3 1 1 2 3 3 2 2 1 3 1	1 2 2 2 2

Example explanation

На рисунке a) изображены функции до добавления вызовов. На рисунках от b) до f) показаны последовательные добавления вызовов функций. Белый круг с черной обводкой обозначает функцию без побочных эффектов, черный "— с побочными эффектами, стрелка изображает вызов функции.

Сначала добавляется рекурсивный вызов функции 1 из самой себя, от чего, очевидно, ответ не изменяется. После добавления вызова функции 3 из функции 2 в программе становится две функции с побочными эффектами: 2 и 3, поскольку функция 3 изначально обладала побочными эффектами. Последующие добавления вызовов не увеличивают количество функций с побочными эффектами.