Задача 1. Цапли

Имя входного файла:	`herons.in`
Имя выходного файла:	`herons.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Петя и Маша пришли в зоопарк. Больше всего Пете понравились цапли. Он был поражен их способностью спать на одной ноге.

В вольере находятся несколько цапель. Некоторые из них стоят на двух ногах, некоторые — на одной. Когда цапля стоит на одной ноге, то другую ее ногу не видно. Петя пересчитал видимые ноги всех цапель, и у него получилось число a.

Через несколько минут к вольеру подошла Маша. За это время некоторые цапли могли поменять позу, поэтому Петя предложил ей заново пересчитать видимые ноги цапель. Когда Маша это сделала, у нее получилось число b.

Выйдя из зоопарка, Петя с Машей заинтересовались, сколько же всего цапель было в вольере. Вскоре ребята поняли, что однозначно определить это число можно не всегда. Теперь они хотят понять, какое минимальное и какое максимальное количество цапель могло быть в вольере.

Требуется написать программу, которая по заданным числам a и b выведет минимальное и максимальное количество цапель, которое могло быть в вольере.

Формат входного файла

Входной файл содержит два целых числа a и b, разделенных ровно одним пробелом ($1 \le a \le 10^9$, $1 \le b \le 10^9$).

Формат выходного файла

Выведите в выходной файл два целых числа, разделенных пробелом — минимальное и максимальное число цапель, которое могло быть в вольере. Гарантируется, что хотя бы одно количество цапель соответствует условию задачи.

Пример входного и выходного файлов

herons.in	herons.out
3 4	2 3

Пояснения к примеру

В приведенном примере возможны следующие варианты:

1) В вольере две цапли. Когда Петя считал ноги, одна цапля стояла на двух ногах, а другая — на одной. Петя насчитал три ноги. Когда Маша считала ноги, обе цапли стояли на двух ногах, Маша насчитала четыре ноги.

2) В вольере три цапли. Когда Петя считал ноги, все цапли стояли на одной ноге, Петя насчитал три ноги. Когда Маша считала ноги, одна цапля стояла на двух ногах, а еще две — на одной. Маша насчитала четыре ноги.

Система оценивания

Правильные решения для тестов, в которых $1 \le a \le 1000$, $1 \le b \le 1000$, будут оцениваться из 50 баллов.

Задача 2. Круглый стол

Имя входного файла:	`table.in`
Имя выходного файла:	`table.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Возрождая древние традиции английских рыцарей, в одном городе члены школьного клуба любителей информатики каждую неделю собираются за круглым столом и обсуждают результаты последних соревнований.

Руководитель клуба Иван Петрович недавно заметил, что не все ребята активно участвуют в обсуждении. Понаблюдав за несколькими заседаниями клуба, он заметил, что активность члена клуба зависит от того, кто с кем сидит рядом.

В клуб приходят на занятия m мальчиков и n девочек. Иван Петрович заметил, что мальчик активно участвует в обсуждении только тогда, когда непосредственно рядом с ним с обеих сторон от него сидят девочки, а девочка активно участвует в обсуждении только тогда, когда непосредственно рядом с ней с одной стороны от нее сидит мальчик, а с другой — девочка.

Желая сделать заседание клуба как можно более интересным, Иван Петрович решил разместить участников за круглым столом таким образом, чтобы как можно больше членов клуба приняло активное участие в обсуждении.

Требуется написать программу, которая по заданным числам m и n выведет такой способ размещения m мальчиков и n девочек за круглым столом, при котором максимальное количество членов клуба будет активно участвовать в обсуждении.

Формат входного файла

Входной файл содержит два целых числа m и n, разделенных ровно одним пробелом ($0 \le m \le 1000$, $0 \le n \le 1000$, $m + n \ge 3$).

Формат выходного файла

Выходной файл должен содержать строку с расположенными в некотором порядке m символами «`B`» (заглавная латинская буква) и n символами «`G`» (заглавная латинская буква). Символ «`B`» означает мальчика, а символ «`G`» — девочку.

Символы следует расположить в том порядке, в котором нужно разместить членов клуба вокруг стола. Соседние символы соответствуют членам клуба, которые сидят рядом. Рядом сидят также члены клуба, соответствующие первому и последнему символу выведенной строки.

Примеры входных и выходных файлов

table.in	table.out
1 2	BGG
2 2	BGBG

Пояснения к примерам

В первом примере все члены клуба примут активное участие в обсуждении.

Во втором примере мальчики примут активное участие в обсуждении, а девочки нет. В этом примере можно также разместить членов клуба следующим образом: «`BBGG`». В этом случае активное участие в обсуждении примут обе девочки, а мальчики — нет. Разместить всех так, чтобы три или четыре члена клуба приняли активное участие в обсуждении, нельзя.

Задача 3. Поврежденный XML

Имя входного файла:	`xml.in`
Имя выходного файла:	`xml.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Формат XML является распространенным способом обмена данными между различными программами. Недавно программист Иванов написал небольшую программу, которая сохраняет некоторую важную информацию в виде XML-строки.

XML-строка состоит из открывающих и закрывающих тегов.

Открывающий тег начинается с открывающей угловой скобки (<), за ней следует имя тега — непустая строка из строчных букв латинского алфавита, а затем закрывающая угловая скобка (>). Примеры открывающих тегов: `<a>`, `<dog>`.

Закрывающий тег начинается с открывающей угловой скобки, за ней следует прямой слеш (/), затем имя тега — непустая строка из строчных букв латинского алфавита, а затем закрывающая угловая скобка. Примеры закрывающихся тегов: `</a>`, `</dog>`.

XML-строка называется *корректной*, если она может быть получена по следующим правилам:

- Пустая строка является корректной XML-строкой.
- Если A и B — корректные XML-строки, то строка AB, получающаяся приписыванием строки B в конец строки A, также является корректной XML-строкой.
- Если A — корректная XML-строка, то строка <X>A</X>, получающаяся приписыванием в начало A открывающегося тега, а в конец — закрывающегося с таким же именем, также является корректной XML-строкой. Здесь X — любая непустая строка из строчных букв латинского алфавита.

Например, представленные ниже строки:

```
<a></a>
<a><ab></ab><c></c></a>
<a></a><a></a><a></a>
```

являются корректными XML-строками, а такие строки как:

```
<a></b>
<a><b>
<a><b></a></b>
```

не являются корректными XML-строками.

Иванов отправил файл с сохраненной XML-строкой по электронной почте своему коллеге Петрову. Однако, к сожалению, файл повредился в процессе пересылки: ровно один символ в строке заменился на некоторый другой символ.

Требуется написать программу, которая по строке, которую получил Петров, восстановит исходную XML-строку, которую отправлял Иванов.

Формат входного файла

Входной файл содержит одну строку, которая заменой ровно одного символа может быть превращена в корректную XML-строку. Длина строки лежит в пределах от 7 до 1000, включительно. Строка содержит только строчные буквы латинского алфавита и символы «<» (ASCII код 60), «>»(ASCII код 62) и «/»(ASCII код 47).

Строка во входном файле заканчивается переводом строки.

Формат выходного файла

Выходной файл должен содержать корректную XML-строку, которая может быть получена из строки во входном файле заменой ровно одного символа на другой. Если вариантов ответа несколько, можно вывести любой.

Примеры входных и выходных файлов

xml.in	xml.out
<a>	<a>
<a><aa>	<a>
<a><>a>	<a>
<a/	<a>

Система оценивания

Правильные решения для тестов, в которых одна буква заменена на другую букву, оцениваются из 50 баллов.

Задача 4. Игра с числами

Имя входного файла:	`game.in`
Имя выходного файла:	`game.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сегодня на уроке математики Петя и Вася изучали понятие арифметической прогрессии. Арифметической прогрессией с разностью d называется последовательность чисел $a_1, a_2, \ldots, a_k$, в которой разность между любыми двумя последовательными числами равна d. Например, последовательность 2, 5, 8, 11 является арифметической прогрессией с разностью 3.

После урока Петя и Вася придумали новую игру с числами. Игра проходит следующим образом.

В корзине находятся n фишек, на которых написаны различные целые числа $a_1, a_2, \ldots, a_n$. По ходу игры игроки выкладывают фишки из корзины на стол. Петя и Вася делают ходы по очереди, первым ходит Петя. Ход состоит в том, что игрок берет одну фишку из корзины и выкладывает ее на стол. Игрок может сам решить, какую фишку взять. После этого он должен назвать целое число $d \geq 2$ такое, что все числа на выбранных к данному моменту фишках являются членами некоторой арифметической прогрессии с разностью d, не обязательно последовательными. Например, если на столе выложены фишки с числами 2, 8 и 11, то можно назвать число 3, поскольку эти числа являются членами приведенной в начале условия арифметической прогрессии с разностью 3.

Игрок проигрывает, если он не может сделать ход из-за отсутствия фишек в корзине или из-за того, что добавление любой фишки из корзины на стол приводит к тому, что он не сможет подобрать соответствующее число d.

Например, если в корзине имеются числа 2, 3, 5 и 7, то Петя может выиграть. Для этого ему необходимо первым ходом выложить на стол число 3. После первого хода у него много вариантов назвать число d, например он может назвать $d = 3$. Теперь у Васи два варианта хода.

1) Вася может вторым ходом выложить фишку с числом 5 и назвать $d = 2$. Тогда Петя выкладывает фишку с числом 7, называя $d = 2$. На столе оказываются фишки с числами 3, 5 и 7, а в корзине осталась только фишка с числом 2. Вася не может ее выложить, поскольку после этого он не сможет назвать корректное число d. В этом случае Вася проигрывает.
2) Вася может вторым ходом выложить фишку с числом 7 и также назвать, например, $d = 2$. Тогда Петя выкладывает фишку с числом 5, называя также $d = 2$. Вася снова попадает в ситуацию, когда на столе оказываются фишки с числами 3, 5 и 7, а в корзине осталась только фишка с числом 2, и он также проигрывает.

Заметим, что любой другой первый ход Пети приводит к его проигрышу. Если он выкладывает число 2, то Вася отвечает числом 7, и Петя не может выложить ни одной фишки. Если Петя выкладывает фишку с числом 5 или 7, то Вася выкладывает фишку с числом 2, и у Пети также нет допустимого хода.

Требуется написать программу, которая по заданному количеству фишек n и числам на фишках $a_1, a_2, \ldots, a_n$ определяет, сможет ли Петя выиграть вне зависимости от действий Васи, и находит все возможные первые ходы Пети, ведущие к выигрышу.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \leq n \leq 200$).

Вторая строка содержит n различных целых чисел $a_1, a_2, \ldots, a_n$ (для всех i от 1 до n выполняется неравенство $1 \leq a_i \leq 10^5$). Соседние числа разделены ровно одним пробелом.

Формат выходного файла

Первая строка выходного файла должна содержать число k — количество различных первых ходов, которые может сделать Петя, чтобы выиграть. Если Вася может выиграть вне зависимости от действий Пети, строка должна содержать цифру 0.

Во второй строке должно содержаться k различных целых чисел — все выигрышные числа. Будем называть число выигрышным, если, выложив в качестве первого хода фишку, содержащую это число, Петя может выиграть вне зависимости от действий Васи. Соседние числа в строке должны быть разделены ровно одним пробелом.

Примеры входных и выходных файлов

game.in	game.out
4 2 3 5 7	1 3
2 2 4	0

Пояснения к примерам

Первый пример рассматривается в тексте условия этой задачи.

Во втором примере, какую бы фишку не выложил Петя первым ходом, Вася в ответ выкладывает другую фишку, и Петя не может сделать ход из-за отсутствия фишек в корзине.

Система оценивания

Правильные решения для тестов, в которых $1 \le n \le 15$, будут оцениваться из 40 баллов.

Задача 5. Кондиционер

Имя входного файла:	`cond.in`
Имя выходного файла:	`cond.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В офисе, где работает программист Петр, установили кондиционер нового типа. Этот кондиционер отличается особой простотой в управлении. У кондиционера есть всего лишь два управляемых параметра: желаемая температура и режим работы.

Кондиционер может работать в следующих четырех режимах:

- «freeze» — охлаждение. В этом режиме кондиционер может только уменьшать температуру. Если температура в комнате и так не больше желаемой, то он выключается.
- «heat» — нагрев. В этом режиме кондиционер может только увеличивать температуру. Если температура в комнате и так не меньше желаемой, то он выключается.
- «auto» — автоматический режим. В этом режиме кондиционер может как увеличивать, так и уменьшать температуру в комнате до желаемой.
- «fan» — вентиляция. В этом режиме кондиционер осуществляет только вентиляцию воздуха и не изменяет температуру в комнате.

Кондиционер достаточно мощный, поэтому при настройке на правильный режим работы он за час доводит температуру в комнате до желаемой.

Требуется написать программу, которая по заданной температуре в комнате t_{room}, установленным на кондиционере желаемой температуре t_{cond} и режиму работы определяет температуру, которая установится в комнате через час.

Формат входного файла

Первая строка входного файла содержит два целых числа t_{room}, и t_{cond}, разделенных ровно одним пробелом ($-50 \le t_{room} \le 50$, $-50 \le t_{cond} \le 50$).

Вторая строка содержит одно слово, записанное строчными буквами латинского алфавита — режим работы кондиционера.

Формат выходного файла

Выходной файл должен содержать одно целое число — температуру, которая установится в комнате через час.

Примеры входных и выходных файлов

cond.in	cond.out
10 20 heat	20
10 20 freeze	10

Пояснения к примерам

В первом примере кондиционер находится в режиме нагрева. Через час он нагреет комнату до желаемой температуры в 20 градусов.

Во втором примере кондиционер находится в режиме охлаждения. Поскольку температура в комнате ниже, чем желаемая, кондиционер самостоятельно выключается и температура в комнате не поменяется.

Задача 6. Праздничный ужин

Имя входного файла:	`dinner.in`
Имя выходного файла:	`dinner.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рядом с офисом компании, в которой работает программист Джон, открылось новое кафе. Директор компании решил провести там новогодний ужин.

Меню праздничного новогоднего ужина в кафе состоит из k типов блюд. Для каждого типа блюда есть несколько вариантов на выбор. Всего есть a_1 вариантов для первого типа блюда, a_2 вариантов для второго типа блюда, и так далее, a_k вариантов для k-го типа блюда. Всего, таким образом, предлагается $a_1 \times a_2 \times \ldots \times a_k$ различных заказов праздничного ужина.

Всего на ужине будут присутствовать m сотрудников компании. Каждый сотрудник должен заказать ровно один вариант блюда каждого типа на выбор. Таким образом, ужин каждого сотрудника будет состоять из k блюд. Для того чтобы ужин каждого сотрудника компании был уникален, администратор кафе придумал следующую схему. Сотрудники делают заказ ужина из меню один за другим. Каждый сотрудник выбирает k блюд, по одному варианту каждого типа. После выбора заказа из меню, сотрудник указывает один их типов блюд, и выбранный этим сотрудником вариант блюда этого типа больше не предлагается тем сотрудникам, которые делают заказ после него.

Каждый сотрудник компании запомнил, сколько возможных заказов ужина ему было предложено. Выяснилось, что директору, который выбирал первым, было предложено на выбор $n_1 = a_1 \times a_2 \times \ldots \times a_k$ заказов. Тому, кто выбирал вторым, досталось лишь $n_2 < n_1$ заказов, поскольку один из вариантов одного из типов блюд уже не был доступен, и так далее. Джону, который выбирал последним, был предложен выбор лишь из n_m заказов. Джон заинтересовался, а какое количество вариантов каждого типа блюд было на выбор у директора компании.

Требуется написать программу, которая по заданным числам k, m и n_1, n_2, …, n_m выяснит, какое количество вариантов каждого типа блюд изначально предлагалось на выбор.

Формат входного файла

Первая строка входного файла содержит два целых числа k и m, разделенных ровно одним пробелом ($1 \le k \le 20$, $2 \le m \le 100$). Вторая строка содержит m чисел: n_1, n_2, …, n_m (для всех i от 1 до m выполняется неравенство $1 \le n_i \le 10^9$).

Формат выходного файла

Выходной файл должен содержать k чисел: a_1, a_2, …, a_k. Если возможных вариантов решения поставленной задачи несколько, требуется вывести любой. Соседние числа должны быть разделены ровно одним пробелом. Гарантируется, что хотя бы одно решение существует.

Пример входного и выходного файлов

dinner.in	dinner.out
3 3 12 8 4	3 2 2

Пояснения к примеру

События в примере могли развиваться, например, следующим образом.

Исходно количество заказов ужина было равно 3×2×2=12. Директор, выбрав свой заказ, указал блюдо первого типа, поэтому второму сотруднику осталось лишь два варианта блюда первого типа. Количество заказов для него сократилось до 2×2×2 = 8. Он также указал на свое блюдо первого типа, и Джон уже мог выбирать лишь из 1×2×2 = 4 заказов ужина.

Задача 7. Космический кегельбан

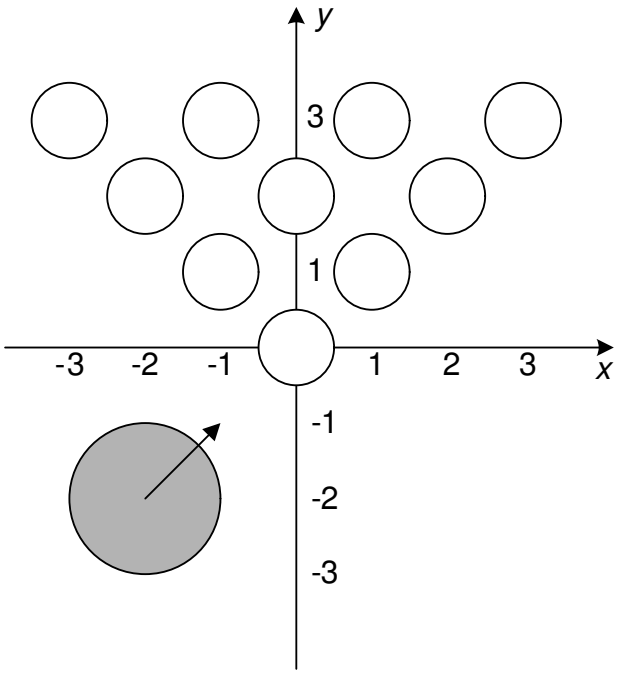
Имя входного файла:	`spacepin.in`
Имя выходного файла:	`spacepin.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

На планете Плюк открылся новый космический кегельбан. Поле для кегельбана представляет собой бесконечную плоскость, на которой расставлены кегли.

Каждая кегля представляет собой высокий цилиндр с основанием в виде круга радиусом r метров. Все кегли одинаковые. Кегли расставлены по следующим правилам. Кегли образуют n рядов, в первом ряду стоит одна кегля, во втором — две, и так далее. В последнем n-м ряду стоит n кеглей. Введем на плоскости систему координат таким образом, чтобы единица измерения была равна одному километру. Центр единственной кегли в первом ряду находится в точке (0, 0). Центры кеглей во втором ряду находятся в точках (–1, 1) и (1, 1). Таким образом, центры кеглей в i-м ряду находятся в точках с координатами $(-(i-1), i-1), (-(i-3), i-1), \ldots, (i-1, i-1)$.

Игра происходит следующим образом. Используется шар с радиусом q метров. Игрок выбирает начальное положение центра шара (x_c, y_c) и вектор направления движения шара (v_x, v_y). После этого шар помещается в начальную точку и двигается, не останавливаясь, в направлении вектора (v_x, v_y). Считается, что шар сбил кеглю, если в процессе движения шара имеет место ситуация, когда у шара и кегли есть общая точка. Сбитые кегли не меняют направления движения шара и не сбивают соседние кегли при падении.

На рисунке приведен пример расположения кеглей для $r = 500$, $n = 4$ и шара для $q = 1000$, $x_c = -2$, $y_c = -2$, $v_x = 1$, $v_y = 1$.



Требуется написать программу, которая по заданным радиусу кегли r, количеству рядов кеглей n, радиусу шара q, его начальному положению (x_c, y_c) и вектору направления движения (v_x, v_y) определяет количество кеглей, сбитых шаром.

Формат входного файла

Первая строка входного файла содержит два целых числа: r и n, разделенных ровно одним пробелом ($1 \le r \le 700$, $1 \le n \le 200\,000$).

Вторая строка входного файла содержит целое число q ($1 \le q \le 10^9$).

Третья строка входного файла содержит два целых числа x_c и y_c, разделенных ровно одним пробелом ($-10^6 \le x_c \le 10^6$, $-10^6 \le y_c$, $1000 \times y_c < -(r + q)$).

Четвертая строка входного файла содержит два целых числа v_x и v_y, разделенных ровно одним пробелом ($-10^6 \le v_x \le 10^6$, $0 < v_y \le 10^6$).

Формат выходного файла

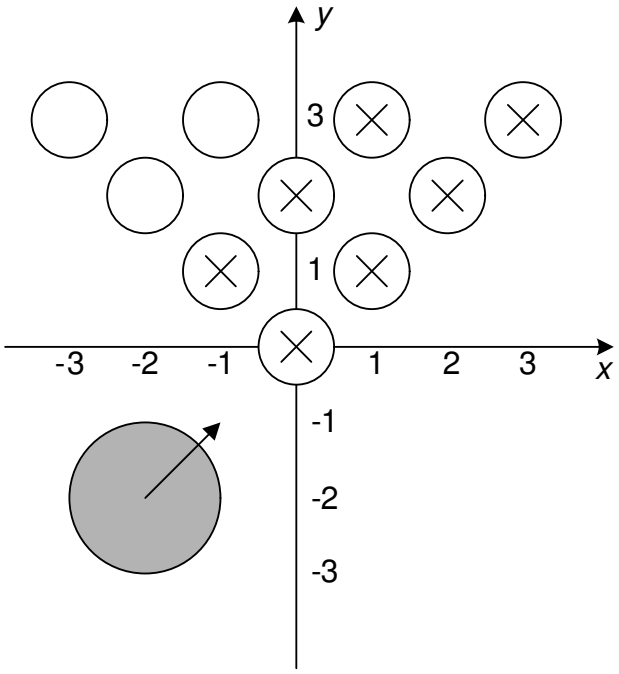
Выходной файл должен содержать одно целое число — количество сбитых кеглей.

Пример входного и выходного файлов

spacepin.in	spacepin.out
500 4 1000 -2 -2 1 1	7

Пояснения к примеру

Рисунок ниже показывает, какие кегли будут сбиты (такие кегли обозначены «x»).



Система оценивания

Правильные решения для тестов, в которых $1 \le n \le 1000$ и $v_x = 0$, будут оцениваться из 20 баллов.

Правильные решения для тестов, в которых $1 \le n \le 1000$ и $v_x \ne 0$, будут оцениваться еще из 20 баллов.

Правильные решения для тестов, в которых $1000 < n \le 200\,000$ и $v_x = 0$, будут оцениваться еще из 20 баллов.

Чтобы получить оставшиеся 40 баллов, решение должно правильно работать также для тестов, в которых $1000 < n \le 200\,000$ и $v_x \ne 0$.

Задача 8. «Abracadabra»

Имя входного файла:	`sufpref.in`
Имя выходного файла:	`sufpref.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Строка s называется *супрефиксом* для строки t, если t начинается с s и заканчивается на s. Например, «abra» является супрефиксом для строки «abracadabra». В частности, сама строка t является своим супрефиксом. Супрефиксы играют важную роль в различных алгоритмах на строках.

В этой задаче требуется решить обратную задачу о поиске супрефикса, которая заключается в следующем. Задан словарь, содержащий n слов t_1, t_2, …, t_n и набор из m строк-образцов s_1, s_2, …, s_m. Необходимо для каждой строки-образца из заданного набора найти количество слов в словаре, для которых эта строка-образец является супрефиксом.

Требуется написать программу, которая по заданному числу n, n словам словаря t_1, t_2, …, t_n, заданному числу m и m строкам-образцам s_1, s_2, …, s_m вычислит для каждой строки-образца количество слов из словаря, для которых эта строка-образец является супрефиксом.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 200\,000$).

Последующие n строк содержат слова t_1, t_2, …, t_n, по одному слову в каждой строке. Каждое слово состоит из строчных букв латинского алфавита. Длина каждого слова не превышает 50. Суммарная длина всех слов не превышает 10^6. Словарь не содержит пустых слов.

Затем следует строка, содержащая целое число m ($1 \le m \le 200\,000$).

Последующие m строк содержат строки-образцы s_1, s_2, …, s_m, по одной на каждой строке. Каждая строка-образец состоит из строчных букв латинского алфавита: Длина каждой строки-образца не превышает 50. Суммарная длина всех строк-образцов не превышает 10^6. Никакая строка-образец не является пустой строкой.

Формат выходного файла

Выходной файл должен содержать m чисел, по одному на строке.

Для каждой строки-образца в порядке, в котором они заданы во входном файле, следует вывести количество слов словаря, для которых она является супрефиксом.

Пример входного и выходного файлов

sufpref.in	sufpref.out
4 abacaba abracadabra aa abra 3 a abra abac	4 2 0

Система оценивания

Правильные решения для тестов, в которых $1 \le n \le 100$, $1 \le m \le 100$, будут оцениваться из 30 баллов.