Задача A. Вычисления в C*++

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Язык C*++ похож на C++. Это сходство проявляется в том, что программы на C*++ порой ведут себя непредсказуемо и приводят к совершенно неожиданным эффектам. Например, представим себе арифметическое выражение на C*++, имеющее следующий вид:

- $expression ::= summand \mid expression + summand \mid expression - summand$
- $summand ::= increment \mid coefficient*increment$
- $increment ::= \text{a++} \mid \text{++a}$
- $coefficient ::= 0|1|2|...|1000$

Например, `5*a++-3*++a+a++` — это корректное арифметическое выражение на C*++.

Итак, мы имеем сумму, состоящую из нескольких слагаемых, разделенных знаками «+» или «-». Каждое слагаемое представляет собой выражение «a++» или «++a», умноженное на некоторый целый коэффициент. Если коэффициент опущен, то предполагается, что он равен 1.

Вычисление такой суммы в C*++ происходит следующим образом. Сначала по очереди вычисляются все слагаемые, после этого происходит суммирование по обычным арифметическим правилам. Если слагаемое содержит «a++», то при его вычислении сначала происходит умножение значения переменной «a» на коэффициент, затем значение «a» увеличивается на 1. Если слагаемое содержит «++a», то действия над ним выполняются в обратном порядке: сначала — увеличение «a» на 1, затем — умножение на коэффициент.

Порядок вычисления слагаемых может быть любым, поэтому иногда результат вычисления выражения совершенно непредсказуем! Ваша задача — найти наибольшее возможное его значение.

Формат входного файла

В первой строке входного файла задано одно целое число a ($-1000 \le a \le 1000$) — начальное значение переменной «a». В следующей строке содержится выражение на языке C*++ описанного вида. Количество слагаемых в выражении не превосходит 1000. Гарантируется, что строка, описывающая выражение, не содержит пробелов и знаков табуляции.

Формат выходного файла

Выведите единственное число — наибольшее возможное значение выражения.

Примеры

stdin	stdout
1 5*a++-3*++a+a++	11
3 a+++++a	8

Note

Рассмотрим второй пример. Сначала $a = 3$. Пусть сначала вычисляется первое слагаемое, потом — второе. Первое слагаемое получается равным 3, и значение a увеличивается на 1. При вычислении второго слагаемого значение a сначала еще раз увеличивается (становится равным 5). Значение второго слагаемого — 5, и в сумме получаем 8. При вычислении сначала второго, а потом первого слагаемого получаем, что оба слагаемых равны 4 и ответ тоже 8.

Задача B. Рост прибыли компании

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Петя работает PR-менеджером процветающей берляндской компании BerSoft. Ему нужно подготовить презентацию о росте прибыли своей компании с 2001 года (года ее основания) до настоящего времени. Пете известно, что в 2001 году прибыль компании составила a_1 млрд. бурлей, в 2002 году — a_2 млрд., ..., в нынешнем $(2000 + n)$-м году — a_n млрд. бурлей. На основании имеющейся информации Петя задумал отразить в своей презентации линейную динамику роста компании, являющуюся, по его мнению, идеальной. Согласно уже построенному Петей графику, в первый год прибыль компании BerSoft должна составлять 1 млрд. бурлей, во второй год — 2 млрд. бурлей и т.д., в каждый следующий год прибыль увеличивается на 1 млрд. бурлей. К сожалению, реальные цифры отличаются от идеальных. Среди чисел a_i даже могут встречаться отрицательные, свидетельствующие об убытках компании в некоторые годы. Поэтому Петя хочет пренебречь некоторыми данными, иначе говоря, вычеркнуть некоторые числа a_i из последовательности и оставить только некоторую подпоследовательность, имеющую идеальный рост.

Таким образом, Пете нужно выбрать такую последовательность годов $y_1, y_2, \ldots, y_k$, чтобы в год y_1 прибыль компании составляла 1 млрд. бурлей, в год y_2 — 2 млрд. бурлей, и т.д., в соответствии с идеальной динамикой роста. Помогите ему выбрать наибольшую такую последовательность.

Формат входного файла

В первой строке содержится целое число n $(1 \le n \le 100)$. В следующей строке заданы n целых чисел a_i $(-100 \le a_i \le 100)$. Число a_i определяет прибыль компании BerSoft в $(2000 + i)$-м году. Числа в строке разделены пробелами.

Формат выходного файла

Выведите k — наибольшую возможную длину идеальной последовательности. В следующей строке выведите саму последовательность годов $y_1, y_2, \ldots, y_k$. Числа разделяйте пробелами. Если решений несколько, выведите любое. Если решения не существует, выведите одно число 0.

Примеры

stdin	stdout
10 -2 1 1 3 2 3 4 -10 -2 5	5 2002 2005 2006 2007 2010
3 -1 -2 -3	0

Задача C. Кратеры на Луне

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 megabytes

Существуют различные теории происхождения кратеров на Луне. Большинство ученых придерживается метеоритной теории, согласно которой кратеры возникли при столкновении небесных тел с Луной. Другая версия — это вулканическое происхождение кратеров.

Специалист по исследованию внеземного разума профессор Окулов (однофамилец того самого Окулова — автора известных учебников по программированию) выдвинул альтернативную гипотезу. Как вы думаете, какую? Правильно, связанную с вмешательством внеземного разума. Теперь профессор увлечен поиском подтверждений своей гипотезы.

В распоряжении профессора имеются данные с лунохода, который движется прямолинейно в одном направлении по поверхности Луны. Кратеры имеют форму кругов с целочисленными радиусами. Луноход фиксирует только те кратеры, центры которых лежат на траектории его движения и отправляет на Землю информацию о расстоянии центров кратеров до начальной точки своего движения и их радиусах.

Согласно теории профессора Окулова, два кратера, созданные внеземным разумом для пока неразгаданных целей, либо полностью вкладываются один в другой, либо не пересекаются вообще. Касание кратеров как внутренним, так и внешним образом допустимо. Однако экспериментальные данные с лунохода не подтверждают эту теорию! Тем не менее профессор Окулов не унывает: он прекрасно понимает, что для создания любой стройной теории необходимо пренебречь некоторыми данными, неверными из-за ошибок измерений (или из-за умелой маскировки внеземного разума, который рано или поздно будет обнаружен профессором Окуловым!). Поэтому Окулов хочет выбрать среди имеющихся описаний кратеров наибольший набор, удовлетворяющий его теории.

Формат входного файла

В первой строке содержится целое число n ($1 \le n \le 2000$) — количество обнаруженных кратеров. Следующие n строк содержат описания кратеров в формате «c_i r_i», где c_i — координата центра кратера на прямой движения лунохода, r_i — радиус кратера. Все числа c_i и r_i — целые положительные, не превосходящие 10^9. Никакие два кратера не совпадают.

Формат выходного файла

В первой строке выведите количество кратеров в искомом наибольшем наборе. В следующей строке через пробел выведите номера входящих в него кратеров. Кратеры нумеруются числами от 1 до n в порядке задания во входных данных. Номера разрешается выводить в произвольном порядке. Если решений несколько, выведите любое.

Примеры

stdin	stdout
4 1 1 2 2 4 1 5 1	3 1 2 4

Задача D. Кубическая планета

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Чего только не встретишь в нашей Галактике! Вокруг икосаэдрической звезды вращается кубическая планета. Введем систему координат таким образом, чтобы ребра кубической планеты были параллельны осям координат, а две противоположные вершины лежали в точках $(0, 0, 0)$ и $(1, 1, 1)$. На планете живут две мухи. В данный момент они сидят в двух различных вершинах кубической планеты. Ваша задача — определить, видят ли они друг друга. Мухи видят друг друга, если занимаемые ими вершины принадлежат одной грани куба.

Формат входного файла

В первой строке записаны три целых числа (0 или 1) через пробел — координаты первой мухи, во второй строке аналогичным образом заданы координаты второй мухи.

Формат выходного файла

Выведите «YES» (без кавычек), если мухи видят друг друга, и «NO» (без кавычек) в противном случае.

Примеры

stdin	stdout
0 0 0 0 1 0	YES
1 1 0 0 1 0	YES
0 0 0 1 1 1	NO

Задача E. При чем тут Дирихле?

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Всем вам известен принцип Дирихле, идея которого в том, что размещение по n коробкам не менее $n+1$ предметов влечет существование коробки, в которой хотя бы два предмета.

Узнав об этом принципе, но не владея техникой логических рассуждений, восьмилетние Стас с Машей придумали игру. Имеется a различимых коробок и b различимых предметов, за ход можно либо добавить новую коробку, либо — новый предмет. Проигрывает в игре тот игрок, после хода которого число способов разложить по a коробкам b предметов становится не меньше некоторого заданного числа n. Все коробки и предметы считаются различными. Возможно, некоторые коробки останутся пустыми.

Кто проиграет при оптимальной игре обоих игроков, если первым ходит Стас?

Формат входного файла

В единственной строке входного файла записано три целых числа a, b, n ($1 \le a \le 10000$, $1 \le b \le 30$, $2 \le n \le 10^9$) — начальное число коробок, предметов и число, ограничивающее количество способов, соответственно. Гарантируется, что изначальное количество способов строго меньше n.

Формат выходного файла

Выведите «Stas», если победит Маша. Выведите «Masha», если победит Стас. Если будет ничья, то выведите «Missing».

Примеры

stdin	stdout
2 2 10	Masha
5 5 16808	Masha
3 1 4	Stas
1 4 10	Missing

Note

Во втором примере первоначальное количество способов равно 3125.

- Если Стас увеличит число коробок, то проиграет, так как Маша следующим ходом может еще раз увеличить число коробок. После этого любой ход Стаса приведет к поражению.
- Если же Стас увеличит число предметов, то любой машин ход будет проигрышным.

Задача F. Лягушки-пацифисты

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

С Дюймовочкой приключилась беда. Она оказалась на маленьком островке посреди болота и очень хочет выбраться на берег.

Выбраться из болота можно только по кочкам, которые расположены вдоль прямой линии, соединяющей островок с берегом. Будем считать, что кочки пронумерованы числами от 1 до n и номер кочки соответствует расстоянию в метрах от островка до нее. Расстояние от n-й кочки до берега тоже 1 метр.

Дюймовочка слишком маленькая, чтобы прыгать на такие расстояния. К счастью, ей предложила помощь семья лягушек, обитающих в болоте. Каждая из них согласна подвезти Дюймовочку, но Дюймовочка должна выбрать только одну лягушку. Любая лягушка имеет определенную длину прыжка. Если Дюймовочка согласится на помощь лягушки с длиной прыжка d, то лягушка прыгнет с островка на кочку с номером d, далее — на кочку с номером $2d$, затем $3d$, и так до тех пор, пока не выберется на берег (т.е. окажется дальше n-й кочки).

Однако есть еще одна проблема: в болоте также обитают комары. Сейчас у них тихий час, и они спят, разместившись на некоторых кочках. Если лягушка прыгнет на кочку с комаром, то она раздавит его. Встретившиеся Дюймовочке лягушки по своим убеждениям пацифисты, поэтому их сильно огорчит гибель любого комара. Помогите Дюймовочке выбрать такую лягушку, которая доставит ее на берег и при этом задавит как можно меньше комаров.

Формат входного файла

В первой строке заданы три целых числа n, m и k ($1 \le n \le 10^9$, $1 \le m, k \le 100$) — количество кочек, лягушек и комаров соответственно. Во второй строке содержатся m целых чисел d_i ($1 \le d_i \le 10^9$) — длины прыжков лягушек. В третьей строке записаны k целых чисел — номера кочек, на которых спит каждый комар. На одной кочке может спать не более одного комара. Числа в строках разделены одиночными пробелами.

Формат выходного файла

В первой строке выведите количество лягушек, которые задавят наименьшее число комаров, во второй — их номера в порядке возрастания, разделенные пробелами. Лягушки номеруются числами от 1 до m в порядке задания длин их прыжков во входных данных.

Примеры

stdin	stdout
5 3 5 2 3 4 1 2 3 4 5	2 2 3
1000000000 2 3 2 5 999999995 999999998 999999996	1 2

Задача G. Обратная функция

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	5 seconds
Ограничение по памяти:	64 megabytes

Петя написал программу на C++, вычисляющую значение одной интересной функции $f(n)$. Петя запустил программу при некотором значении n и отправился на кухню пить чай. О том, сколько времени работала программа, история умалчивает. Когда Петя вернулся, она уже закончила выполнение и выдала результат. Однако пока Петя пил чай, коварный вирус успел уничтожить входной файл, поэтому Петя не может восстановить, для какого значения n была запущена программа. Помогите Пете: реализуйте обратную функцию!

Основную часть программы представляет собой функция на C++ со следующим упрощенным синтаксисом:

- *function* ::= int f(int n) {*operatorSequence*}
- *operatorSequence* ::= *operator* | *operator operatorSequence*
- *operator* ::= return *arithmExpr*; | if (*logicalExpr*) return *arithmExpr*;
- *logicalExpr* ::= *arithmExpr* > *arithmExpr* | *arithmExpr* < *arithmExpr* | *arithmExpr* == *arithmExpr*
- *arithmExpr* ::= *sum*
- *sum* ::= *product* | *sum* + *product* | *sum* − *product*
- *product* ::= *multiplier* | *product* ∗ *multiplier* | *product*/*multiplier*
- *multiplier* ::= n | *number* | f(*arithmExpr*)
- *number* ::= 0|1|2|...|32767

Пробелы и переводы строк в *operatorSequence* — необязательны.

Таким образом, имеется функция, в теле которой встречается два вида операторов. Это оператор “return *arithmExpr*;”, возвращающий в качестве значения функции значение арифметического выражения, и условный оператор “if (*logicalExpr*) return *arithmExpr*;”, который возвращает значение арифметического выражения в том и только в том случае, когда выполняется логическое выражение. Гарантируется, что никакие другие конструкции языка C++: циклы, операторы присваивания, вложенные условные операторы и т.д. и другие переменные, кроме параметра n, в функции не используются. Все константы являются целыми числами из промежутка [0..32767].

Операторы выполняются последовательно. После того, как произойдет возврат функцией некоторого значения, остальные операторы в последовательности не выполняются. Арифметические выражения вычисляются с учетом стандартного приоритета операций. Это означает, что сначала вычисляются все произведения, входящие в сумму. При вычислении произведения операции умножения и деления выполняются слева направо. Затем происходит суммирование слагаемых, при этом сложение и вычитание тоже выполняются слева направо. Операции ”>” (больше), ”<” (меньше) и ”==” (равно) тоже имеют стандартный смысл.

Теперь внимание! Программа компилируется с помощью разработанного берляндской компанией BerSoft 15-битного компилятора Berland C++, поэтому арифметические действия выполняются нестандартным образом. Сложение, вычитание и умножение выполняются по модулю 32768 (если при вычитании получается отрицательное число, то к нему прибавляется 32768 до тех пор, пока оно не окажется в промежутке [0..32767]). Деление ”/” — это обычное целочисленное деление с отбрасыванием остатка.

Примеры арифметических действий:

$$12345 + 23456 = 3033, \quad 0 - 1 = 32767, \quad 1024 * 1024 = 0, \quad 1000/3 = 333.$$

Гарантируется, что при любом значении n от 0 до 32767 заданная функция выполняется корректно. Это означает, что:

1. Не происходит деления на 0.

2. При выполнении функции для значения $n = N$ рекурсивные вызовы функции f могут происходить только для значений параметра $0, 1, \ldots, N-1$. Следовательно, программа никогда не уходит в бесконечную рекурсию.

3. В итоге выполнения последовательности операторов обязательно происходит возврат значения функции.

Заметим, что в силу всех введенных ограничений значение, возвращаемое функцией f, не зависит ни от глобальных переменных, ни от порядка вычисления арифметических выражений в составе логического, ни от чего либо другого, кроме значения параметра n. Поэтому функцию f можно рассматривать как функцию в математическом смысле, т.е. как однозначное соответствие каждому значению n из промежутка [0..32767] значения $f(n)$ из того же промежутка.

Вам дано значение $f(n)$, нужно найти n. Если подходящих значений n несколько, требуется определить максимальное (из промежутка [0..32767]).

Формат входного файла

В первой строке содержится целое число $f(n)$ из промежутка [0..32767]. В следующих строках содержится описание функции f. В описании могут встречаться дополнительные пробелы и переводы строки (см. примеры), которые, разумеется, не могут разрывать ключевые слова int, if, return и числа. Размер входных данных не превосходит 100 байт.

Формат выходного файла

Выведите единственное число — ответ на задачу. Если решения не существует, выведите "-1" (без кавычек).

Примеры

stdin	stdout

17
int f(int n)
{
if (n < 100) return 17;
if (n > 99) return 27;
}
	99

13
int f(int n)
{
if (n == 0) return 0;
return f(n - 1) + 1;
}
	13

144
int f(int n)
{
if (n == 0) return 0;
if (n == 1) return n;
return f(n - 1) + f(n - 2);
}
	24588

Задача H. Таблица умножения

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Петя изучает позиционные системы счисления. Он уже научился складывать и вычитать числа в системах счисления с различными основаниями и теперь перешел к более сложному действию — умножению. Для того, чтобы умножать большие числа, нужно сначала выучить таблицу умножения. К сожалению, во втором классе (а некоторые даже в первом) учат только таблицу умножения десятичных чисел. Помогите Пете построить таблицу умножения для чисел в системе счисления с основанием k.

Формат входного файла

В первой строке содержится единственное целое число k ($2 \le k \le 10$) — основание системы счисления.

Формат выходного файла

Выведите таблицу умножения для системы счисления с основанием k. Таблица должна содержать $k-1$ строку и $k-1$ столбец. Элемент на пересечении i-й строки и j-го столбца равен произведению чисел i и j в k-ичной системе счисления. Между числами в каждой строке может содержаться произвольное количество пробелов (в примерах лишние пробелы выведены для наглядности).

Примеры

stdin	stdout
10	1 2 3 4 5 6 7 8 9 2 4 6 8 10 12 14 16 18 3 6 9 12 15 18 21 24 27 4 8 12 16 20 24 28 32 36 5 10 15 20 25 30 35 40 45 6 12 18 24 30 36 42 48 54 7 14 21 28 35 42 49 56 63 8 16 24 32 40 48 56 64 72 9 18 27 36 45 54 63 72 81
3	1 2 2 11

Задача I. Трамвай

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

В берляндском городе C*** есть трамвайное депо и всего один трамвай. В депо работают три человека: водитель трамвая, кондуктор и начальник депо. Раньше трамвай каждое утро выезжал из депо и ездил по круговому маршруту. При этом трамвай проходил маршрут ровно за c минут. Начальник депо контролировал движение трамвая, выходя на улицу через каждые c минут, когда трамвай проезжал мимо депо, и лишал водителя трамвая премии, если тот опаздывал хотя бы на секунду.

Так было в прежние времена. Впоследствии из федерального бюджета Берляндии были выделены средства на расширение сети трамвайных путей в городе C***, и, как иногда бывает, средства были использованы по назначению. Сеть трамвайных путей была перестроена, в результате чего превратилась в обширную и разветвленную систему. Возможно, прежний круговой маршрут был разрушен. В городе C*** n перекрестков и теперь m трамвайных путей, соединяющих пары перекрестков. Движение в Берляндии одностороннее, поэтому по каждому из путей трамвай может ехать в одном направлении. Между двумя перекрестками может проходить несколько трамвайных путей, с одним или разными направлениями движения. Каждый путь соединяет пару различных перекрестков и имеет такую длину, что трамвай проходит его ровно за 1 минуту. Для каждого перекрестка существует хотя бы один путь, по которому можно уехать с этого перекрестка.

Итак, трамвайные пути были построены, но почему-то никто не позаботился об увеличении количества трамваев в городе C***! Трамвай так и продолжал ездить в одиночестве, но теперь у водителя появилась прекрасная возможность избавиться от неусыпного контроля начальника депо. Ведь благодаря разветвленной сети он получил свободу в выборе маршрута! Теперь водитель, достигнув очередного перекрестка, может выбрать путь, по которому поедет, произвольным образом. При этом трамвай может уехать даже в такие районы города C***, откуда невозможно вернуться обратно в депо из-за однонаправленного движения. Водитель не боится этой трудности: ночью, когда город спит, можно незаметно вернуться в депо, двигаясь по путям в обратном направлении.

Горожане ликовали. Ведь некоторые из них ждали трамвая на своих улицах вот уже несколько лет. Однако поведение водителя привело в ярость начальника депо. Теперь он вынашивает коварный план установки видеонаблюдения за непокорным трамваем.

План заключается в следующем. Начальник депо хочет установить видеокамеры на некоторых перекрестках, выбрать промежуток времени t и через каждые t минут отвлекаться от любимой телепередачи, чтобы проверить местоположение трамвая по камерам. При этом начальник депо хочет, чтобы во все моменты времени, кратные t, и только в них трамвай оказывался на перекрестке под камерой наблюдения. На перекрестке, где находится депо, камера должна быть установлена обязательно, с целью предотвращения возможного теракта против начальника депо. Среди всех возможных планов начальник депо выбирает план с наибольшим значением t (уж очень не любит он отвлекаться от любимой телепередачи, но приходится). Если таких планов несколько, выбирается тот, для которого требуется как можно меньше камер наблюдения. Найдите такой план.

Формат входного файла

В первой строке содержатся целые числа n и m ($2 \le n, m \le 10^5$) — количество перекрестков и количество трамвайных путей в городе C*** соответственно. Следующие m строк содержат описания путей в формате «u v», где u — начальный перекресток пути, а v — его конечный перекресток. Перекрестки пронумерованы целыми числами от 1 до n, причем трамвайное депо находится на перекрестке с номером 1.

Формат выходного файла

В первой строке выведите значение t. В следующей строке выведите число k — необходимое количество камер наблюдения. В следующей строке через пробел выведите номера перекрестков,

на которых следует установить камеры. Номера выводите в порядке возрастания.

Примеры

stdin	stdout
4 5 1 2 2 3 3 4 4 1 1 4	2 2 1 3

Задача J. Проверка правописания

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 megabytes

Петя заметил, что когда он набирает текст на клавиатуре, у него часто нажимаются лишние клавиши и в словах возникают лишние буквы. Конечно же, система проверки правописания подчеркивает ему эти слова, ему приходится кликать на слово и выбирать правильный вариант. Пете надоело исправлять свои ошибки вручную, поэтому он решил реализовать функцию, которая сама будет вносить исправления. Петя начал с разбора наиболее часто встречающегося у него случая, когда из слова достаточно удалить одну букву, чтобы оно совпало с некоторым словом из словаря. Итак, Петя столкнулся с такой подзадачей: дано введенное слово и слово из словаря, нужно удалить из первого слова одну букву, чтобы получилось второе. И тогда перед Петей встал весьма нетривиальный вопрос: какую букву удалять?

Формат входного файла

Входные данные содержат две строки, состоящие из строчных латинских букв. Длина строк от 1 до 10^6 символов включительно, первая строка содержит ровно на 1 символ больше, чем вторая.

Формат выходного файла

В первой строке выведите количество позиций символов в первой строке, при удалении каждого из которых получается вторая строка. Во второй строке через пробел выведите сами позиции в порядке возрастания. Позиции нумеруются с 1. Если из первой строки невозможно получить вторую путем удаления одного символа, выведите одно число 0.

Примеры

stdin	stdout
abdrakadabra abrakadabra	1 3
aa a	2 1 2
competition codeforces	0

Задача K. Испытания

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Вы участвуете в испытаниях нового оружия. Для испытаний создан полигон, представляющий собой прямоугольное поле размера $n \times m$, разделенное на единичные квадраты 1×1. На полигоне расположено k объектов, каждый из которых является прямоугольником со сторонами, параллельными сторонам полигона, занимающим полностью несколько единичных квадратов. Объекты не пересекаются и не касаются друг друга.

Принцип действия оружия сверхсекретен. Вам известно лишь то, что с помощью него можно нанести удар по любой прямоугольной области ненулевой площади со сторонами, параллельными сторонам полигона. Область должна накрывать полностью некоторые из единичных квадратов, на которые разбит полигон, и никак не затрагивать остальные квадраты. Разумеется, область не должна выходить за границы полигона. При этом вам поставлена задача: поразить не менее одного и не более трех прямоугольных объектов. Каждый объект должен лежать либо целиком внутри области (тогда он считается пораженным), либо целиком вне области.

Посчитайте количество способов нанести удар.

Формат входного файла

В первой строке содержатся три целых числа n, m и k ($1 \le n, m \le 1000$, $1 \le k \le 90$) — размеры полигона и количество объектов на нем, соответственно. Следующие n строк содержат по m символов каждая и описывают полигон. Символ «*» означает, что соответствующий квадрат на полигоне занят объектом, символ «.» обозначает пустое пространство. Гарантируется, что символы «*» образуют ровно k связных прямоугольных областей, отвечающих условию задачи.

Формат выходного файла

Выведите единственное число — количество способов нанести удар.

Примеры

stdin	stdout
3 3 3 *.* ... *..	21
4 5 4 .*.** ...** **... ...**	38
2 2 1 .* ..	4