Задача A. Алхимия

Имя входного файла:	`alchemy.in`
Имя выходного файла:	`alchemy.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Алхимки Средневековья владели знаниями о превращении различных химических веществ друг в друга. Это подтверждают и недавние исследования археологов.

В ходе археологических раскопок было обнаружено m глиняных табличек, каждая из которых была покрыта непонятными на первый взгляд символами. В результате расшифровки выяснилось, что каждая из табличек описывает одну алхимическую реакцию, которую умели проводить алхимики.

Результатом алхимической реакции является превращение одного вещества в другое. Заданы набор алхимических реакций, описанных на найденных глиняных табличках, исходное вещество и требуемое вещество. Необходимо выяснить, возможно ли преобразовать исходное вещество в требуемое с помощью этого набора реакций, а в случае положительного ответа на этот вопрос — найти минимальное количество реакций, необъодимое для осуществления такого преобразования.

Формат входного файла

Первая строка входного файла содержит целое число m $(0 \le m \le 1000)$ — количество записей в книге.

Каждая из последующих m строк описывает одну алхимическую реакцию и имеет формат `вещество1 -> вещество2`, где `вещество1` — название исходного вещества, `вещество2` — название продукта алхимической реакции.

$m+2$-ая строка входного файла содержит название вещества, которое имеется исходно, $m+3$-ая — название вещества, которое требуется получить.

Во входном файле упоминается не более 100 различных веществ. Название каждого из веществ состоит из строчных и заглавных латинских букв и имеет длину не более 20 символов. Строчные и заглавные буквы различаются.

Формат выходного файла

В выходной файл выведите минимальное количество алхимических реакций, которое требуется для получения требуемого вещества из исходного, или -1, если требуемое вещество невозможно получить.

Примеры

<table>
<tr><th>alchemy.in</th><th>alchemy.out</th></tr>
<tr><td><pre>5
Aqua -> AquaVita
AquaVita -> PhilosopherStone
AquaVita -> Argentum
Argentum -> Aurum
AquaVita -> Aurum
Aqua
Aurum</pre></td><td><pre>2</pre></td></tr>
<tr><td><pre>5
Aqua -> AquaVita
AquaVita -> PhilosopherStone
AquaVita -> Argentum
Argentum -> Aurum
AquaVita -> Aurum
Aqua
Osmium</pre></td><td><pre>-1</pre></td></tr>
</table>

Задача B. Скобочки

Имя входного файла:	`brackets.in`
Имя выходного файла:	`brackets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Напомним, что называется ıправильной скобочной последовательностью:

- пустая строка является правильной скобочной последовательностью;
- если строка a — правильная скобочная последовательность, то строки (a), $\{a\}$, $[a]$ — тоже правильные скобочные последовательности;
- если строки a и b — правильные скобочные последовательности, то строка ab — тоже правильная скобочная последовательность.

Циклическим сдвигом строки s называется строка $s_k s_{k+1} s_{k+2} \ldots s_{|s|} s_1 s_2 \ldots s_{k-1}$ для некоторого k, где $|s|$ — длина строки s.

Вам дана скобочная последовательность s — строка, состоящая из символов `{`, `}`, `(`, `)`, `[`, `]`. Выясните, является ли s циклическим сдвигом правильной скобочной последовательности.

Формат входного файла

В первой строке входного файла записана строка s (длина строки s не превышает 1000).

Формат выходного файла

Если s является циклическим сдвигом правильной скобочной последовательностью, выведите "YES", иначе — выведите "NO".

Примеры

brackets.in	brackets.out
`}()[]{`	YES
`}([)]{`	NO
`()][`	YES

Задача C. Наилучший делитель

Имя входного файла:	divisorb.in
Имя выходного файла:	divisorb.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Будем говорить, что число a *лучше* числа b, если сумма цифр a больше суммы цифр числа b, а в случае равенства сумм их цифр, если число a меньше числа b. Например, число 124 лучше числа 123, так как у первого из них сумма цифр равна семи, а у второго — шести. Также, число 3 лучше числа 111, так как у них равны суммы цифр, но первое из них меньше.

Дано число n. Найдите такой его делитель (само число n и единица считаются делителями числа n), который лучше любого другого делителя числа n.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 10^5$).

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

divisorb.in	divisorb.out
10	5
239	239

Задача D. Наихудший делитель

Имя входного файла:	`divisorw.in`
Имя выходного файла:	`divisorw.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Как и в предыдущей задаче, будем говорить, что число a *лучше* числа b, если сумма цифр a больше суммы цифр числа b, а в случае равенства сумм их цифр, если число a меньше числа b. Например, число 124 лучше числа 123, так как у первого из них сумма цифр равна семи, а у второго — шести. Также, число 3 лучше числа 111, так как у них равны суммы цифр, но первое из них меньше.

Дано число n. Найдите такой его делитель d (само число n и единица считаются делителями числа n), что любой другой делитель c числа n лучше, чем d.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 10^{5000}$).

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

divisorw.in	divisorw.out
10	10
239	1

Задача E. Число e

Имя входного файла:	`enumber.in`
Имя выходного файла:	`enumber.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Выведите в выходной файл округленное до n знаков после десятичной точки число e. Число e, округленное до 25-ти знаков после десятичной точки, равно 2.7182818284590452353602875.

Формат входного файла

Первая строка входного файла содержит целое число n ($0 \le n \le 25$).

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

enumber.in	**enumber.out**
0	3
25	2.7182818284590452353602875
13	2.7182818284590

Задача F. Карта

Имя входного файла:	`map.in`
Имя выходного файла:	`map.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Одним из разделов математического анализа является теория сжимающих операторов. Важным фактом, который в ней доказывается, является теорема Банаха. Она гласит, что у оператора сжатия есть ровно одна неподвижная точка.

Интересным следствием из этой теоремы является следующее утверждение. Пусть есть карта небольшой части поверхности Земли (поверхность считается плоской). Если карту положить в некотором месте той части поверхности, которую она изображает, то будет существовать ровно одна точка, изображение которой на карте лежит на ней.

Для удобства будем считать, что изображенная на карте часть поверхности Земли имеет форму прямоугольника со сторонами $2W$ и $2H$ метров. Введем прямоугольную декартову систему координат так, что ось Ox направлена с запада на восток, а ось Oy — с юга на север. Единичный отрезок выберем равным одному метру. Кроме этого, поместим начало координат в центр рассматриваемой части поверхности Земли, а стороны рассматриваемого прямоугольника параллельны осям координат. Расположим карту размером $2a$ на $2b$ сантиметров так, что ее центр находится в точке с координатами (x, y). Таким образом, изображенная на карте поверхность Земли имеет форму прямоугольника с углами $(W, H), (-W, H), (-W, -H), (W, -H)$, а углы карты расположены в точках $(x + \frac{a}{100}, y + \frac{b}{100}), (x - \frac{a}{100}, y + \frac{b}{100}), (x - \frac{a}{100}, y - \frac{b}{100}), (x + \frac{a}{100}, y - \frac{b}{100})$.

Найдите точку, изображение которой лежит на ней при таком расположении карты.

Формат входного файла

Входной файл содержит целые числа W, H, x, y, a, b ($1 \le W, H, x, y, a, b \le 1000$). Гарантируется, что карта целиком лежит внутри той части поверхности Земли, которая на ней изображена.

Формат выходного файла

В выходной файл выведите координаты искомой точки с точностью до 10^{-6}.

Примеры

map.in	**map.out**
`10 10 0 0 5 5`	`0.0 0.0`
`10 10 1 1 5 10`	`1.0050251256281408` `1.0101010101010102`

Задача G. Шаблоны

Имя входного файла:	`patterns.in`
Имя выходного файла:	`patterns.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Шаблоном размера n назовем строку длины n, каждый из символов которой входит в множество $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, ?\}$. Шаблоны преобразуются в строки из цифр по следующим правилам:

- символы от 0 до 9 могут быть преобразованы только сами в себя;
- символ a может преобразован в любой из символов $0, 1, 2, 3$;
- символ b может преобразован в любой из символов $1, 2, 3, 4$;
- символ c может преобразован в любой из символов $2, 3, 4, 5$;
- символ d может преобразован в любой из символов $3, 4, 5, 6$;
- символ e может преобразован в любой из символов $4, 5, 6, 7$;
- символ f может преобразован в любой из символов $5, 6, 7, 8$;
- символ g может преобразован в любой из символов $6, 7, 8, 9$;
- символ ? может преобразован в любой из символов от 0 до 9;

Даны два шаблона: p_1 и p_2. Рассмотрим множество S_1 строк, которые могут быть получены из p_1 по описанным правилам, и множество S_w строк, которые могут быть получены из p_2. Необходимо найти количество строк, входящих в оба этих множества.

Формат входного файла

Первая строка входного файла содержит шаблон p_1, вторая — шаблон p_2. Шаблоны имеют одинаковый положительный размер, не превосходящий 9.

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

patterns.in	**patterns.out**
??? abc	64
??? 000	1
abc 999	0

Задача H. Гонки

Имя входного файла:	`rally.in`
Имя выходного файла:	`rally.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В области L находится n городов. Некоторые пары городов соединены проселочной дорогой с двусторонним движением. Начавшись в каком-то городе, дорога не может закончиться в нем же. В этом году состояние дорог позволило отделению ГИБДД области L провести гонки под лозунгом «Скажем НЕТ нарушениям скоростного режима». Было решено, что круговая трасса должна состоять из четырех дорог, но не может проходить через один город два раза. Естественно, свернуть с одной дороги на другую можно только в городе. Организаторы уже должны приступить к составлению отчета, и для этого требуется посчитать количество различных трасс.

Формат входного файла

В первой строке входного файла записаны количество городов n ($1 \le n \le 300$) и количество дорог m. В каждой из следующих m строк содержится два различных числа — номера городов, соединенных соответствующей дорогой.

Формат выходного файла

В выходной файл выведите одно число — количество круговых трасс из четырех дорог, которые могут составить организаторы.

Примеры

rally.in	**rally.out**
4 6 1 2 2 3 3 4 4 1 1 3 2 4	3

Задача I. Строки

Имя входного файла:	`strings.in`
Имя выходного файла:	`strings.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Циклическим сдвигом строки s называется строка $s_k s_{k+1} s_{k+2} \ldots s_{|s|} s_1 s_2 \ldots s_{k-1}$ для некоторого k, здесь $|s|$ — длина строки s.

Подстрокой строки s называется строка $s_i s_{i+1} \ldots s_{j-1} s_j$ для некоторых i и j.

Вам даны две строки a и b. Выведите количество подстрок строки a, являющихся циклическими сдвигами строки b.

Формат входного файла

В первой строке входного файла записана строка a ($1 \le |a| \le 1000$). Во второй строке входного файла записана строка b ($1 \le |b| \le \min(100, |a|)$).

Обе строки состоят только из символов латинского алфавита и цифр.

Формат выходного файла

В первой строке выходного файа выведите одно целое число — ответ на задачу.

Примеры

strings.in	**strings.out**
abcabc abc	4
abcabc acb	0
aaaaaaa aa	6
aAaa8aaAa aAa	4