# Задача A. Byteland Ice

| | |
|---|---|
| Имя входного файла: | `bytelandice.in` |
| Имя выходного файла: | `bytelandice.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

Компания «Byteland Ice» открыла в столице Байтландии сеть кафе, торгующих мороженым. Мороженое продаётся порциями от одного до трёх шариков.

При этом ценовая политика компании такова, что, если покупатель берёт порцию в три шарика мороженого, то цена за шарик будет меньше, чем в случае, если он берёт два, а если он берёт два, то меньше, чем при покупке единственного шарика. От сорта мороженого цена порции не зависит.

Компания из нескольких студентов-любителей мороженого решила при покупке мороженого использовать этот факт, сделав «экономный» заказ: например, если три человека берут по одному шарику, то заказывается порция из трёх шариков и распределяется между ними.

В кафе на тот момент было мороженое двух сортов: шоколадное и клубничное. Если покупатель заказывает порцию (двойную или тройную), в которой есть два шарика разного сорта, то из-за таяния мороженого у каждого из шариков появляется дополнительный привкус (у клубничного — шоколадный и наоборот). Тем, кто собирается пробовать мороженое обоих сортов, это не мешает, но для тех, кто планирует заказать мороженое только одного какого-то сорта, важно, чтобы доставшиеся им шарики не были куплены «в комплекте» с другим сортом.

По заданным ценам на обычную, двойную и тройную порции и списку заказов каждого из студентов найдите минимальную сумму, за которую можно выполнить все эти заказы.

## Формат входного файла

В первой строке входного файла заданы четыре целых числа: $n$, $s$, $d$ и $t$, где $n$ — количество студентов, а $s$, $d$ и $t$ — цена порции мороженого из одного, двух и трёх шариков соответственно ($1 \leqslant n \leqslant 100,\ 1 \leqslant s < d < t \leqslant 1000,\ s > d/2 > t/3$). В последующих $n$ строках задано количество порций мороженого, заказанных каждым студентом: $c$ — количество порций шоколадного мороженого и $r$ — количество порций клубничного мороженого соответственно ($0 \leqslant r, c \leqslant 10^4$).

## Формат выходного файла

Выведите одно целое число — минимальную стоимость общего заказа.

## Примеры

| bytelandice.in | bytelandice.out |
|---|---|
| 1 300 400 500<br>1 1 | 400 |
| 2 6 8 9<br>1 0<br>0 2 | 14 |
| 3 24 32 42<br>0 2<br>3 1<br>1 1 | 116 |

# Задача B. Group

| | |
|---|---|
| Имя входного файла: | `group.in` |
| Имя выходного файла: | `group.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

В математике *группой* называется объект, состоящий из множества $G$ и оператора $\times$ таких, что

- Если $a$ и $b$ принадлежат $G$, то $a \times b$ принадлежит G.
- Если $a$, $b$ и $c$ принадлежат $G$, то $(a \times b) \times c = a \times (b \times c)$.
- Существует элемент $i$ такой, что $a \times i = i \times a = a$.
- Для каждого элемента $a$ существует обратный элемент $a^{-1}$ такой, что $a \times a^{-1} = a^{-1} \times a = i$.

Пусть нам задано множество из $n$ элементов (для простоты пронумеруем эти элементы цифрами от 1 до $n$). «Таблицей умножения» для данного множества и оператора $\times$ назовём таблицу из $n$ строк и $n$ столбцов, в которой на пересечении $j$-й строки и $k$-го столбца стоит $j \times k$. По заданной таблице умножения проверьте, является ли соответствующая ей пара «множество, оператор» группой.

## Формат входного файла

В первой строке входного файла задано целое число $n$ ($1 \leqslant n \leqslant 100$) — количество элементов в множестве.

Далее следует $n$ строк по $n$ целых чисел $p_{jk}$ в каждой. $k$-е число в $j$-й строке обозначает элемент, полученный в результате операции $j \times k$ ($1 \leqslant p_{jk} \leqslant 100$).

## Формат выходного файла

Выведите "`Yes`", если заданная таблица умножения определяет группу, и "`No`" в противном случае.

## Примеры

| group.in | group.out |
|---|---|
| 6<br>1 2 3 4 5 6<br>2 1 1 1 1 1<br>3 1 1 1 1 1<br>4 1 1 1 1 1<br>5 1 1 1 1 1<br>6 1 1 1 1 1 | No |
| 2<br>1 3<br>3 1 | No |
| 2<br>2 1<br>1 2 | Yes |

# Задача C. Nicknames

| | |
|---|---|
| Имя входного файла: | `nicknames.in` |
| Имя выходного файла: | `nicknames.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

Как известно, во многих социальных сетях появляются «двойники» популярных никнеймов — имена, внешне очень похожие на них. Администрация, что естественно, борется с этим явлением. В одной социальной сети администрация решила пойти ещё дальше и в связи с годом Дракона запретила все имена, которые являются префиксами или суффиксами существующих имён, либо для которых существующие имена являются префиксами или суффиксами.

При реализации драконовских мер произошёл сбой, и база пользователей обнулилась. Так что три пользователя, решившие зарегистрироваться сразу после сбоя, оказались на тот момент единственными пользователями сети.

Ваша задача — выяснить, получится ли у них зарегистрироваться под желаемыми никами.

## Формат входного файла

В первой строке входного файла задано число $N$ ($1 \leqslant N \leqslant 5$)— количество тестовых примеров. Далее следуют сами тестовые примеры. Каждый пример состоит из трёх непустых строк, состоящих из не более 25 строчных латинских букв каждая.

## Формат выходного файла

Для каждой тестовой строки выведите “`Yes`”, если у пользователей получится зарегистрироваться под желаемыми никами, и “`No`”, если система проверки префиксов и суффиксов не допустит этого.

## Пример

| nicknames.in | nicknames.out |
|---|---|
| 2<br>xyyx<br>xxy<br>yxy<br>z<br>zt<br>zz | Yes<br>No |

# Задача D. Triangles and circles

| | |
|---|---|
| Имя входного файла: | `tricir.in` |
| Имя выходного файла: | `tricir.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

Задано некоторое количество треугольников и окружностей. Необходимо разбить их на пары так, чтобы в каждой паре был ровно один треугольник и ровно одна окружность. При этом для каждой пары должно выполняться условие: треугольник можно поместить внутрь окружности так, чтобы все его точки лежали внутри или на границе окружности. Определите максимальное число возможных пар.

## Формат входного файла

В первой строке входного файла записано целое число $N$ ($0 < N \leqslant 1000$) — число различных объектов. В каждой из следующих $N$ строк дано описание одного объекта — либо окружности, либо треугольника.

Окружность описывается строкой в формате "`1 R`", где целое $R$ — радиус окружности ($0 < R \leqslant 1000$).

Треугольник описывается строкой в формате "`2 A B C`", где целые $A$, $B$, $C$ — длины сторон треугольника ($0 < A, B, C \leqslant 1000$), гарантируется что треугольник невырожденный и существует.

## Формат выходного файла

Выведите максимальное число пар «треугольник-окружность», удовлетворяющих условию задачи.

## Примеры

| tricir.in | tricir.out |
|---|---|
| 2<br>1 3<br>2 3 4 6 | 1 |
| 6<br>1 2<br>1 3<br>1 4<br>2 3 3 4<br>2 4 4 5<br>2 5 5 6 | 2 |

# Задача E. Vugluscr on chessboard

| | |
|---|---|
| Имя входного файла: | `vugluscr.in` |
| Имя выходного файла: | `vugluscr.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

Из шахматной доске $M \times N$ вырезано несколько клеток. В некоторых из оставшихся клеток расставлены цифры от 1 до 9 — количество очков, которые даются за прохождение через эту клетку.

В левом нижнем углу доски находится вуглускр. Вуглускр — это фигура, обладающая следующими свойствами: во-первых, он ходит только вправо, вверх или вниз на соседнюю по стороне клетку (если та существует), во-вторых, если он сделал ход вверх или вниз, то следующий ход он обязан сделать в том же направлении или вправо (то есть последовательности ходов «вниз-вверх» и «вверх-вниз» невозможны). В правом нижнем углу клетки сидит мышь. Требуется провести вуглускра к мыши, набрав максимальное количество очков. Гарантируется, что существует хотя бы один путь от вуглускра до мыши (это значит, в частности, что левая нижняя и правая нижняя клетки гарантированно существуют).

## Формат входного файла

В первой строке входного файла заданы два целых числа $M$ и $N$ ($1 \leqslant M, N \leqslant 100$) — соответственно высота и ширина доски. Далее в $M$ строках, содержащих по $N$ символов каждая, задана сама доска. Символ '`.`' (точка) обозначает свободную клетку, символ '`*`' — вырезанную клетку, цифры от 1 до 9 — свободную клетку, за прохождение через которую даётся соответствующее количество очков.

## Формат выходного файла

Выведите одно число — максимальное количество очков, которое можно набрать, проведя вуглускра к мыши.

## Пример

| vugluscr.in | vugluscr.out |
|---|---|
| 6 10<br>..2.......<br>..........<br>..6.**....<br>.8**...1..<br>..7..9....<br>.......... | 25 |
| 2 2<br>22<br>11 | 6 |

# Задача F. Old classical [:|||:]

| | |
|---|---|
| Имя входного файла: | `wolf.in` |
| Имя выходного файла: | `wolf.out` |
| Ограничение по времени: | 2 seconds |
| Ограничение по памяти: | 256 mebibytes |

Головоломка “Волк, коза и капуста” формулируется следующим образом:

Крестьянину нужно перевести через реку с левого берега на правый волка, козу и капусту. Но лодка такова, что в ней может поместиться крестьянин, а с ним или только волк, или только коза, или только капуста. Но если оставить волка с козой, то волк съест козу, а если оставить козу с капустой, то коза съест капусту. Как перевезти свой груз крестьянину?

Перед вами та же самая задача, но немного усложненная. Предположим, что вначале крестьянин, волк, коза и капуста могут находиться на любом берегу — на правом или на левом. Задача остается прежней: перевезти всех на правый берег без потерь. Определите, как это можно сделать. Учтите, что лодка без крестьянина плыть не может.

## Формат входного файла

В первой строке входного файла через пробел указаны объекты, которые находятся на левом берегу, во второй строке — объекты, которые находятся на правом берегу. Приняты следующие обозначения: “`Farmer`” — крестьянин, “`Wolf`” — волк, “`Goat`” — коза, “`Cabbage`” — капуста. В случае, если на каком-то берегу объектов нет, соответствующая строка остаётся пустой (**в том числе и в конце файла**)

## Формат выходного файла

Если решения нет, то выведите “`Impossible`”. В противном случае выведите (возможно, пустую) последовательность объектов (каждый следующий — в новой строке), перевозя которые с берега на берег, удастся решить головоломку (выводите “`Self`” в случае, если крестьянин переплывает реку без других объектов в лодке, см. пример). Если решений несколько - выведите любое.

## Примеры

| wolf.in | wolf.out |
|---|---|
| Farmer Cabbage Goat<br>Wolf | Cabbage<br>Self<br>Goat |
| Farmer<br>Goat Cabbage Wolf | Impossible |