Задача A. Упорядоченные кубики

У Пети есть набор кубиков. Строго говоря, не каждый кубик является кубом, но является прямоугольным параллелепипедом. Петя полагает, что кубики можно сравнивать между собой по их измерениям (длине, ширине и высоте). Так, кубик A считается меньшим чем кубик B, если можно их так расположить относительно друг друга, что их соответствующие ребра будут параллельны и при этом ширина A будет меньше ширины B, длина A — меньше длины B и высота A — меньше высоты B.

Рассмотрим пример. Измерения (длина x ширина x высота) кубика A: $3 \times 3 \times 2$. Измерения кубика B: $4 \times 3 \times 4$. Можно повернуть кубик A так, чтобы его измерения стали $3 \times 2 \times 3$ и убедиться, что он меньше кубика B. Если добавить кубик C с измерениями $5 \times 2 \times 1$, можно видеть, что он не больше и не меньше кубика A и кубика B.

Петя разложил свой набор кубиков в последовательность $C_1, C_2, \ldots C_N$. Найдите в этой последовательности самую длинную цепочку кубиков, упорядоченных по возрастанию, то есть набор $C_K, C_{K+1}, \ldots C_{K+M}$, где кубик C_K меньше C_{K+1}, C_{K+1} меньше C_{K+2}, и т. д., а K и M таковы, что M наибольшее из возможных. Цепочка может состоять из одного кубика. Если искомых цепочек несколько, следует указать ту, что находится ближе к концу последовательности.

Формат входных данных

На вход программа принимает натуральное число N ($1 \leqslant N \leqslant 100'000$) — количество кубиков. В последующих N строках ввода программе заданы измерения кубиков — тройки натуральных чисел a_i, b_i, c_i, $0 < a_i, b_i, c_i \leqslant 10'000$.

Формат выходных данных

В первой строке необходимо вывести длину искомой цепочки кубиков. В последующих строках измерения каждого из кубиков цепочки. Измерения следует выводить в том виде, в котором они были даны во вводе.

Примеры

тест	ответ
5 2 1 1 3 2 2 3 1 3 4 4 2 5 3 5	3 3 1 3 4 4 2 5 3 5

Задача B. Бармаглот

Компания «Barmaley's Computing» учла пожелания пользователей и выпустила вторую версию компьютера — Barmaglot-2. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер остался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека могут содержать либо число, либо латинскую букву или пробел, либо специальное значение `[BARMALEY]`, которое используется для обозначения логической лжи, при том что любое другое обозначает истину.

Программа для Barmaglot'а представляет собой строку символов, каждый из которых задаёт машинную команду; программа, состоящая из символов-команд, выполняется последовательно слева направо, кроме двух команд, которые могут нарушить эту последовательность.

- Команды 'a', 'b', 'c' и все остальные латинские буквы означают "занести данную букву в регистр".
- Команды '0', '1', ..., '9' означают "занести в регистр соответствующее число".
- Команда '@' означает занести в регистр пробел.
- Команды '+', '-', '*', '/', '%' означают соответствующие арифметические действия, операции целочисленные. % - операция взятия остатка
- Команды '<', '>', '=' означают сравнение двух чисел или двух символов, '&' и '|' означают логическое "и"и логическое "или"; все эти команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.
- Команда '#' умножает содержимое регистра в десять раз,
- Команда '_' делит содержимое регистра в десять раз.

- Команда '!' работает как логическое отрицание содержимого регистра: если там значение [BARMALEY], то заносится значение 1, если любое другое - заносится значение [BARMALEY].
- Команда '.' выдаёт текущее значение регистра на печать
- команда '?' вводит очередное число(целиком, а не по разрядам), а если на вводе кончилась (оборвалась) лента, заносит в регистра значение [BARMALEY].
- Команда ']' заносит значение из регистра в стек;
- Команда '[' извлекает значение с вершины стека и заносит его в регистра (если извлекать нечего, в регистр заносится [BARMALEY]);
- Команда '~' меняет местами значения в регистра и на вершине стека.
- Команда '}' заносит значение из регистра в очередь,
- Команда '{' извлекает из очереди самое старое значение и помещает в аккумулятор (или помещает туда [BARMALEY], если очередь пуста).
- Команды '(' и ')' предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Команда (, если в регистре [BARMALEY], идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в регистре было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда), наоборот, если в регистре [BARMALEY], не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной круглой скобки, после чего продолжает выполнение с команды, стоящей после такой скобки справа.

Наконец, команда '"' прекращает выполнение программы, при этом выполнение считается успешным. Если программа кончилась, не встретив эту команду, она завершается аварийно.

В вашем распоряжении оказался эмулятор данного компьютера (http://ejudge.cs.msu.ru/barmaglot2/) и требуется написать для него программу, решающую следующую задачу:

В одной компьютерной игре игрок выставляет в линию шарики разных цветов. Когда образуется непрерывная цепочка из трех и более шариков одного цвета, она удаляется из линии. Все шарики при этом сдвигаются друг к другу, и ситуация может повториться.

Напишите программу, которая по данной ситуации определяет, сколько шариков будет сейчас "уничтожено". Естественно, непрерывных цепочек из трех и более одноцветных шаров в начальный момент может быть не более одной.

Формат входных данных

Сначала вводится количество шариков в цепочке (не более 1000) и цвета шариков (от 0 до 9, каждому цвету соответствует свое целое число).

Формат выходных данных

Требуется вывести количество шариков, которое будет "уничтожено".

Примеры

тест	ответ
5 1 3 3 3 2	3

Задача C. Корпорация добра

В одной очень известной фирме существует четкая иерархическая структура. У каждого из N работников есть свой непосредственный начальник. Директором фирмы является сотрудник под номером 0, у него нет начальника.

Руководство фирмы приняло решение об увольнении некоторых служащих и пометило их в специальном списке. К счастью, не все еще решено окончательно. Если разозлить любого сотрудника, то он вычеркивает из списка на увольнение всех своих подчиненных, которые были в этом списке и дописывает в него всех своих подчиненных, которых там не было. (Когда сотрудник злится, он становится настолько злым, что считает своими подчиненными себя, своих непосредственных подчиненных, непосредственных подчиненных своих непосредственных подчиненных и так далее)

Борис, как невероятно добрый человек решил спасти ситуацию. Он решил разозлить некоторых сотрудников так, чтобы никого в итоге не пришлось увольнять. Помогите Борису узнать минимальное количество сотрудников, которое ему придется злить.

Формат входных данных

В первой строке вводится число N ($1 \leqslant N \leqslant 10000$) — количество сотрудников в очень известной фирме. В следующих $N-1$ строках в i-й строке вводится непосредственный начальник i-го сотрудника.

Далее вводится число M ($0 \leqslant M \leqslant 10000$) — количество сотрудников в изначальном списке на увольнение. Далее вводится M различных целых чисел A_i — список номеров сотрудников в изначальном списке на увольнение.

Формат выходных данных

Выведите одно число - минимальное количество сотрудников, которое придется злить Борису.

Примеры

тест	ответ
3 0 1 2 1 2	1

Задача D. Нарисованный граф

Петя очень любит рисовать графы в тетрадке в клетку. В углах клеток могут быть нарисованы вершины графа, некоторые соседние вершины могут быть соединены ребрами, при этом ребра могут быть только длины 1 (сторона одной клетки). Нарисовав граф, Петя нумерует вершины и выписывает список ребер.

Однажды Петя потерял листочек с нарисованным графом, но у него остался список ребер. Помогите ему восстановить рисунок.

Формат входных данных

В этой задаче у вас есть несколько входных файла в тестирующей системе, для каждого из которых требуется отправить на проверку файл с графом. В первой строке исходного файла находятся число вершин N и число ребер M. В следующих строках пары числе a, b, означающие наличие ребра между вершиной с номером a и b.

Гарантируется что граф во входном файле всегда задает какой-то из нарисованных графов.

Формат выходных данных

В файле с ответом граф имеет следующий формат.

- Вершина обозначается символом `'o'`
- Ребра обозначаются символами `'-'` или `'|'`
- Ребро имеет всегда длину 1 и состоит из одного символа
- Пустые места обозначаются символом `'.'`

Примеры

тест	ответ
4 4 1 2 2 3 3 4 4 1	o-o |.| o-o

тест	ответ
6 6 1 2 2 3 3 4 4 1 1 5 3 6	o-o-o.. ..|.|.. ..o-o-o

Задача E. Распознавание языка

В тестирующей системе вам дан архив с исходными текстами программ и файлом (files.txt), который содержит перечисление исходных кодов с указанием языка программирования (c++, pascal, python и т.д.)

Вам требуется написать программу, которой на стандартный поток передается список из какого-то подмножества файлов в архиве. Для каждого файла из списка требуется определить язык программирования.

Например, если в исходном архиве две программы в файлах file1

```
int quest(int x) {
    while (x < 1024) {
        for(int y = 0; y < x; y = y + 1) {
            if (y * y == x) {
                return y;
            }
        }
        x = x + 1;
    }
    return -1;
}
```

и file2

```
def f():
    for i in range(10):
        print(i)
```

то в файле files.txt будут следующие строки:

```
2
file1 c++
file2 python
```

Вашей программе при проверке будет передан на стандартный поток ввода список файлов, имена которых могут отличаться, но содержимое каждого совпадает с содержимым какого-то из файлов в архиве. Ваша задача по заданному файлу определить язык программирования, который соответствует языку данного файла из архива. Например, если содержимое somefile совпадает с содержимым file2, то это язык python.

Формат входных данных

На стандартном потоке задается число файлов N ($1 \leqslant N \leqslant 100$), для которых мы хотим определить язык, затем в N строках имена файлов. Гарантируется что, вашей программе доступны эти файлы для открытия на чтение.

Формат выходных данных

Для каждого файла выведите язык программирования.

Примеры

тест	ответ
1 somefile	python

Задача F. Оптимизация программы

Как известно, программы на таких языках, как Си++, Паскаль и др., как правило, необходимо перед запуском скомпилировать. Результатом компиляции является исполняемый файл, содержащий команды процессора, которые будут исполняться при запуске программы. Исполняемый файл является специфичным для операционной системы и архитектуры процессора. Например, исполняемый файл для Windows и архитектуры x64 не может быть напрямую запущен на операционной системе iOS и процессоре ARM.

Когда необходимо работать с командами процессора обычно используется не бинарное представление, хранимое в исполняемом файле, а язык ассемблера. Язык ассемблера позволяет записывать команды процессора в человеко-читаемой форме. Мы рассмотрим некоторое упрощения языка ассемблера, достаточное для решения задачи.

Процессор исполняет команды последовательно начиная от первой команды в файле и до конца файла. Каждая команда в файле занимает одну строку текста и имеет следующий вид:

```
[LABELS] OPCODE [ARG]
```

`LABELS` — это необязательная последовательность *меток*, каждая метка - это цепочка заглавных и строчных латинских букв, цифр и знаков `.` (точка), `_`, `$`. Метка завершается знаком «двоеточие». Перед метками, между метками и знаком двоеточния, после двоеточия могут располагаться пробельные символы (то есть пробел или табуляция). Примеры:

```
L1: L2:
.L4$:
 L3 :
```

`OPCODE` — это операция процессора. Определены следующие операции:

- `C` — операция сравнения. Операция сравнения устанавливает *флаги результата* в зависимости от результата операции над своими аргументами.
- `A` — операция вычисления. Операция не модифицирует флаги результата, которые, возможно, остались после предыдущей операции сравнения.
- `Z` — операция вычисления, в отличие от `A` эта операция модифицирует флаги результата непредсказуемым образом.
- `E` — последняя операция в программе. Эта операция всегда является текстуально последней. Она всегда единственная в файле.
- `B` — операция передачи управления.

Операции `C`, `A`, `Z` имеют единственный аргумент в виде `#ID`, где `ID` — это положительное целое 32-битное число. По сути — это уникальный идентификатор каждой команды в программе. Ни один `ID` не используется в программе дважды. Операция `E` не имеет аргументов. Операция `B` имеет один аргумент — метку.

Рассмотрим программу:

```
        A       #1
        A       #2
        B       L1
L2:     C       #3
        Z       #4
        B       L3
L1:     Z       #5
        B       L2
L3:     A       #6
        E
```

Операция `B L1` передает управление на метку `L1`, то есть после этой инструкции следующей будет выполняться инструкция `B L2`. С учетом таких передач управления рассмотренный фрагмент эквивалентен следующему:

```
        A       #1
        A       #2
        Z       #5
        C       #3
        Z       #4
        A       #6
        E
```

В исходной программе было 10 команд процессора, а в преобразованной стало 7 команд.

Семантикой программы назовем последовательность идентификаторов команд в том порядке, в котором они выполнялись процессором. В семантику программы не входят аргументы инструкции `B`, хотя они учитываются при построении семантики.

Задача оптимизации программы заключается в нахождении программы минимальной длины, семантика которой совпадает с семантикой исходной программы.

Операции, рассмотренные выше, не позволяют реализовать условные операторы или циклы. Условное выполнение реализуется с помощью суффиксов условного выполнения. Например, команда `B` всегда выполняет переход в другую точку программы (говорят «выполняет безусловный переход»), а команда `BEQ` выполняет переход только если предыдущая инструкция сравнения `C` установила флаг равенства аргументов.

Команда сравнения `C` устанавливает флаги «равно» и «больше», что позволяет проверять аргументы на равенство, неравенство, больше, меньше, больше или равно, меньше или равно. Соответствующие суффиксы условного выполнения записываются как `EQ` (equal), `NE` (not equal), `GT` (greater than), `LT` (less than), `GE` (greater or equal), `LE` (less or equal). Суффиксы условного выполнения могут записываться после любой инструкции, кроме `E`. Однако команда `Z` устанавливает флаги некоторым непредсказуемым образом, и поэтому после команды `Z` использовать условное выполнение нельзя.

Например,

```
        C       #1
        A       #2
        BEQ     L1      ; переход, если сравнение #1 установило флаг равенства
        BGT     L2      ; переход, если сравнение #1 установило флаг <<больше>>
```

Условно выполняться может любая инструкция. Например, фрагмент программы:

```
        C       #1
        BLT     L1
        A       #2
        B       L2
L1:     A       #3
L2:
```

этот фрагмент программы использует условные переходы, но он может быть оптимизирован следующим образом:

```
        C       #1
        AGE     #2
        ALT     #3
```

Вам нужно скачать архив программ. Архив имеет формат .zip, и файлы с программами называются in01.asm, in02.asm, ..., in10.asm. Вам необходимо оптимизировать выданные вам программы, то есть найти семантически-эквивалентные программы минимальной длины. В преобразованной команде должны быть сохранены все команды 'C', 'A', 'Z', которые присутствовали в исходной программе. Их порядок может быть текстуально другим, но при условии, что при выполнении программы при любом возможном исходе выполнения условной операции порядок выполнения операций 'C', 'A', 'Z' сохранится неизменным.

В качестве ответа вам нужно будет сдать 10 оптимизированных программ.

Оптимизированная программа не засчитывается, если в нем нарушен синтаксис программы, или получившаяся программа не будет эквивалентна исходной программе.

Задача A. Упорядоченные кубики

У Пети есть набор кубиков. Строго говоря, не каждый кубик является кубом, но является прямоугольным параллелепипедом. Петя полагает, что кубики можно сравнивать между собой по их измерениям (длине, ширине и высоте). Так, кубик A считается меньшим чем кубик B, если можно их так расположить относительно друг друга, что их соответствующие ребра будут параллельны и при этом ширина A будет меньше ширины B, длина A — меньше длины B и высота A — меньше высоты B.

Рассмотрим пример. Измерения (длина x ширина x высота) кубика A: $3 \times 3 \times 2$. Измерения кубика B: $4 \times 3 \times 4$. Можно повернуть кубик A так, чтобы его измерения стали $3 \times 2 \times 3$ и убедиться, что он меньше кубика B. Если добавить кубик C с измерениями $5 \times 2 \times 1$, можно видеть, что он не больше и не меньше кубика A и кубика B.

Петя разложил свой набор кубиков в последовательность $C_1, C_2, \ldots C_N$. Найдите в этой последовательности самую длинную цепочку кубиков, упорядоченных по убыванию, то есть набор $C_K, C_{K+1}, \ldots C_{K+M}$, где кубик C_K больше C_{K+1}, C_{K+1} больше C_{K+2}, и т. д., а K и M таковы, что M наибольшее из возможных. Цепочка может состоять из одного кубика. Если искомых цепочек несколько, следует указать ту, что находится ближе к концу последовательности.

Формат входных данных

На вход программа принимает натуральное число N ($1 \leqslant N \leqslant 100'000$) — количество кубиков. В последующих N строках ввода программе заданы измерения кубиков — тройки натуральных чисел a_i, b_i, c_i, $0 < a_i, b_i, c_i \leqslant 10'000$.

Формат выходных данных

В первой строке необходимо вывести длину искомой цепочки кубиков. В последующих строках измерения каждого из кубиков цепочки. Измерения следует выводить в том виде, в котором они были даны во вводе.

Примеры

тест	ответ
5 2 1 1 3 2 2 5 3 5 4 4 2 3 1 3	3 5 3 5 4 4 2 3 1 3

Задача B. Бармаглот

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Компания «Barmaley's Computing» учла пожелания пользователей и выпустила вторую версию компьютера — Barmaglot-2. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер остался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека могут содержать либо число, либо латинскую букву или пробел, либо специальное значение `[BARMALEY]`, которое используется для обозначения логической лжи, при том что любое другое обозначает истину.

Программа для Barmaglot'a представляет собой строку символов, каждый из которых задаёт машинную команду; программа, состоящая из символов-команд, выполняется последовательно слева направо, кроме двух команд, которые могут нарушить эту последовательность.

- Команды 'a', 'b', 'c' и все остальные латинские буквы означают "занести данную букву в регистр".
- Команды '0', '1', ..., '9' означают "занести в регистр соответствующее число".
- Команда '@' означает занести в регистр пробел.
- Команды '+', '-', '*', '/', '%' означают соответствующие арифметические действия, операции целочисленные. % - операция взятия остатка
- Команды '<', '>', '=' означают сравнение двух чисел или двух символов, '&' и '|' означают логическое "и"и логическое "или"; все эти команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.
- Команда '#' умножает содержимое регистра в десять раз,
- Команда '_' делит содержимое регистра в десять раз.

- Команда '!' работает как логическое отрицание содержимого регистра: если там значение [BARMALEY], то заносится значение 1, если любое другое - заносится значение [BARMALEY].
- Команда '.' выдаёт текущее значение регистра на печать
- команда '?' вводит очередное число(целиком, а не по разрядам), а если на вводе кончилась (оборвалась) лента, заносит в регистра значение [BARMALEY].
- Команда ']' заносит значение из регистра в стек;
- Команда '[' извлекает значение с вершины стека и заносит его в регистра (если извлекать нечего, в регистр заносится [BARMALEY]);
- Команда '~' меняет местами значения в регистра и на вершине стека.
- Команда '}' заносит значение из регистра в очередь,
- Команда '{' извлекает из очереди самое старое значение и помещает в аккумулятор (или помещает туда [BARMALEY], если очередь пуста).
- Команды '(' и ')' предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Команда (, если в регистре [BARMALEY], идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в регистре было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда), наоборот, если в регистре [BARMALEY], не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной круглой скобки, после чего продолжает выполнение с команды, стоящей после такой скобки справа.

Наконец, команда '"' прекращает выполнение программы, при этом выполнение считается успешным. Если программа кончилась, не встретив эту команду, она завершается аварийно.

В вашем распоряжении оказался эмулятор данного компьютера (http://ejudge.cs.msu.ru/barmaglot2/) и требуется написать для него программу, решающую следующую задачу:

В одной компьютерной игре игрок выставляет в линию шарики разных цветов. Когда образуется непрерывная цепочка из трех и более шариков одного цвета, она удаляется из линии. Все шарики при этом сдвигаются друг к другу, и ситуация может повториться.

Напишите программу, которая по данной ситуации определяет, сколько шариков останется. Естественно, непрерывных цепочек из трех и более одноцветных шаров в начальный момент может быть не более одной.

Формат входных данных

Сначала вводится количество шариков в цепочке (не более 1000) и цвета шариков (от 0 до 9, каждому цвету соответствует свое целое число).

Формат выходных данных

Требуется вывести количество шариков, которые останутся.

Примеры

тест	ответ
5 1 3 3 3 2	2

Задача C. Корпорация добра

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

В одной очень известной фирме существует четкая иерархическая структура. У каждого из N работников есть свой непосредственный начальник. Директором фирмы является сотрудник под номером 0, у него нет начальника.

Руководство фирмы приняло решение об увольнении некоторых служащих и пометило их в специальном списке. К счастью, не все еще решено окончательно. Если разозлить любого сотрудника, то он вычеркивает из списка на увольнение всех своих подчиненных, которые были в этом списке и дописывает в него всех своих подчиненных, которых там не было. (Когда сотрудник злится, он становится настолько злым, что считает своими подчиненными себя, своих непосредственных подчиненных, непосредственных подчиненных своих непосредственных подчиненных и так далее)

Борис — сотрудник компании, соперничающей с очень известной компанией. Он решил разозлить некоторых сотрудников так, чтобы в итоге все сотрудники были уволены. Помогите Борису узнать минимальное количество сотрудников, которое ему придется злить.

Формат входных данных

В первой строке вводится число N ($1 \leqslant N \leqslant 10000$) — количество сотрудников в очень известной фирме. В следующих $N-1$ строках в i-й строке вводится непосредственный начальник i-го сотрудника.

Далее вводится число M ($0 \leqslant M \leqslant 10000$) — количество сотрудников в изначальном списке на увольнение. Далее вводится M различных целых чисел A_i — список номеров сотрудников в изначальном списке на увольнение.

Формат выходных данных

Выведите одно число - минимальное количество сотрудников, которое придется злить Борису.

Примеры

тест	ответ
3 0 1 2 1 2	2

Задача D. Нарисованный граф

Петя очень любит рисовать графы в тетрадке в клетку. В углах клеток могут быть нарисованы вершины графа, некоторые соседние вершины могут быть соединены ребрами, при этом ребра могут быть только длины 1 (сторона одной клетки). Нарисовав граф, Петя нумерует вершины и выписывает список ребер.

Однажды Петя потерял листочек с нарисованным графом, но у него остался список ребер. Помогите ему восстановить рисунок.

Формат входных данных

В этой задаче у вас есть несколько входных файла в тестирующей системе, для каждого из которых требуется отправить на проверку файл с графом. В первой строке исходного файла находятся число вершин N и число ребер M. В следующих строках пары числе a, b, означающие наличие ребра между вершиной с номером a и b.

Гарантируется что граф во входном файле всегда задает какой-то из нарисованных графов.

Формат выходных данных

В файле с ответом граф имеет следующий формат.

- Вершина обозначается символом `'o'`
- Ребра обозначаются символами `'-'` или `'|'`
- Ребро имеет всегда длину 1 и состоит из одного символа
- Пустые места обозначаются символом `'.'`

Примеры

тест	ответ
4 4 1 2 2 3 3 4 4 1	o-o |.| o-o

тест	ответ
6 6 1 2 2 3 3 4 4 1 1 5 3 6	o-o-o.. ..|.|.. ..o-o-o

Задача E. Распознавание языка

В тестирующей системе вам дан архив с исходными текстами программ и файлом (files.txt), который содержит перечисление исходных кодов с указанием языка программирования (c++, pascal, python и т.д.)

Вам требуется написать программу, которой на стандартный поток передается список из какого-то подмножества файлов в архиве. Для каждого файла из списка требуется определить язык программирования.

Например, если в исходном архиве две программы в файлах file1

```
int quest(int x) {
    while (x < 1024) {
        for(int y = 0; y < x; y = y + 1) {
            if (y * y == x) {
                return y;
            }
        }
        x = x + 1;
    }
```

```
    return -1;
}
```

и file2

```
def f():
    for i in range(10):
        print(i)
```

то в файле files.txt будут следующие строки:

```
2
file1 c++
file2 python
```

Вашей программе при проверке будет передан на стандартный поток ввода список файлов, имена которых могут отличаться, но содержимое каждого совпадает с содержимым какого-то из файлов в архиве. Ваша задача по заданному файлу определить язык программирования, который соответствует языку данного файла из архива. Например, если содержимое somefile совпадает с содержимым file2, то это язык python.

Формат входных данных

На стандартном потоке задается число файлов N ($1 \leqslant N \leqslant 100$), для которых мы хотим определить язык, затем в N строках имена файлов. Гарантируется что, вашей программе доступны эти файлы для открытия на чтение.

Формат выходных данных

Для каждого файла выведите язык программирования.

Примеры

тест	ответ
1 somefile	python

Задача F. Оптимизация программы

Как известно, программы на таких языках, как Си++, Паскаль и др., как правило, необходимо перед запуском скомпилировать. Результатом компиляции является исполняемый файл, содержащий команды процессора, которые будут исполняться при запуске программы. Исполняемый файл является специфичным для операционной системы и архитектуры процессора. Например, исполняемый файл для Windows и архитектуры x64 не может быть напрямую запущен на операционной системе iOS и процессоре ARM.

Когда необходимо работать с командами процессора обычно используется не бинарное представление, хранимое в исполняемом файле, а язык ассемблера. Язык ассемблера позволяет записывать команды процессора в человеко-читаемой форме. Мы рассмотрим некоторое упрощения языка ассемблера, достаточное для решения задачи.

Процессор исполняет команды последовательно начиная от первой команды в файле и до конца файла. Каждая команда в файле занимает одну строку текста и имеет следующий вид:

```
[LABELS] OPCODE [ARG]
```

`LABELS` — это необязательная последовательность *меток*, каждая метка - это цепочка заглавных и строчных латинских букв, цифр и знаков `.` (точка), `_`, `$`. Метка завершается знаком «двоеточие». Перед метками, между метками и знаком двоеточния, после двоеточия могут располагаться пробельные символы (то есть пробел или табуляция). Примеры:

```
L1: L2:
.L4$:
 L3 :
```

`OPCODE` — это операция процессора. Определены следующие операции:

- `C` — операция сравнения. Операция сравнения устанавливает *флаги результата* в зависимости от результата операции над своими аргументами.
- `A` — операция вычисления. Операция не модифицирует флаги результата, которые, возможно, остались после предыдущей операции сравнения.
- `Z` — операция вычисления, в отличие от `A` эта операция модифицирует флаги результата непредсказуемым образом.
- `E` — последняя операция в программе. Эта операция всегда является текстуально последней. Она всегда единственная в файле.
- `B` — операция передачи управления.

Операции `C`, `A`, `Z` имеют единственный аргумент в виде `#ID`, где `ID` — это положительное целое 32-битное число. По сути — это уникальный идентификатор каждой команды в программе. Ни один `ID` не используется в программе дважды. Операция `E` не имеет аргументов. Операция `B` имеет один аргумент — метку.

Рассмотрим программу:

```
        A       #1
        A       #2
        B       L1
L2:     C       #3
        Z       #4
        B       L3
L1:     Z       #5
        B       L2
L3:     A       #6
        E
```

Операция `B L1` передает управление на метку `L1`, то есть после этой инструкции следующей будет выполняться инструкция `B L2`. С учетом таких передач управления рассмотренный фрагмент эквивалентен следующему:

```
        A       #1
        A       #2
        Z       #5
        C       #3
        Z       #4
        A       #6
        E
```

В исходной программе было 10 команд процессора, а в преобразованной стало 7 команд.

Семантикой программы назовем последовательность идентификаторов команд в том порядке, в котором они выполнялись процессором. В семантику программы не входят аргументы инструкции `B`, хотя они учитываются при построении семантики.

Задача оптимизации программы заключается в нахождении программы минимальной длины, семантика которой совпадает с семантикой исходной программы.

Операции, рассмотренные выше, не позволяют реализовать условные операторы или циклы. Условное выполнение реализуется с помощью суффиксов условного выполнения. Например, команда `B` всегда выполняет переход в другую точку программы (говорят «выполняет безусловный переход»), а команда `BEQ` выполняет переход только если предыдущая инструкция сравнения `C` установила флаг равенства аргументов.

Команда сравнения `C` устанавливает флаги «равно» и «больше», что позволяет проверять аргументы на равенство, неравенство, больше, меньше, больше или равно, меньше или равно. Соответствующие суффиксы условного выполнения записываются как `EQ` (equal), `NE` (not equal), `GT` (greater than), `LT` (less than), `GE` (greater or equal), `LE` (less or equal). Суффиксы условного выполнения могут записываться после любой инструкции, кроме `E`. Однако команда `Z` устанавливает флаги некоторым непредсказуемым образом, и поэтому после команды `Z` использовать условное выполнение нельзя.

Например,

```
        C       #1
        A       #2
        BEQ     L1      ; переход, если сравнение #1 установило флаг равенства
        BGT     L2      ; переход, если сравнение #1 установило флаг <<больше>>
```

Условно выполняться может любая инструкция. Например, фрагмент программы:

```
        C       #1
        BLT     L1
        A       #2
        B       L2
L1:     A       #3
L2:
```

этот фрагмент программы использует условные переходы, но он может быть оптимизирован следующим образом:

```
        C       #1
        AGE     #2
        ALT     #3
```

Вам нужно скачать архив программ. Архив имеет формат .zip, и файлы с программами называются in01.asm, in02.asm, ..., in10.asm. Вам необходимо оптимизировать выданные вам программы, то есть найти семантически-эквивалентные программы минимальной длины. В преобразованной команде должны быть сохранены все команды 'C', 'A', 'Z', которые присутствовали в исходной программе. Их порядок может быть текстуально другим, но при условии, что при выполнении программы при любом возможном исходе выполнения условной операции порядок выполнения операций 'C', 'A', 'Z' сохранится неизменным.

В качестве ответа вам нужно будет сдать 10 оптимизированных программ.

Оптимизированная программа не засчитывается, если в нем нарушен синтаксис программы, или получившаяся программа не будет эквивалентна исходной программе.