Задача A. Выпуклая оболочка

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

Рассмотрим бесконечный лист клетчатой бумаги. Закрасим некоторое непустое множество клеток в черный цвет. Теперь мы хотим закрасить минимальное количество клеток, так, чтобы множество черных клеток стало выпуклым.

Напомним, что геометрическая фигура Φ называется выпуклой, если для любых точек $A \in \Phi$ и $B \in \Phi$ с вещественными координатами отрезок $[AB] \in \Phi$.

Формат входного файла

В первой строке входного файла содержатся два числа N ($1 \le N \le 100$) и M ($1 \le M \le 100$) — размеры куска бумаги, куда попали все черные клетки. В каждой из следующих N строк содержится M символов «`*`» или «`.`». Символ «*» обозначает черную клетку, а «.» белую.

Формат выходного файла

Выведете выпуклое множество, содержащее минимальное количество дополнительно покрашенных черных клеток, в ровно N строках по M символов «*» или «.» в каждой.

Примеры

convex.in	convex.out
2 4 ..*. .**.	.**. .**.
4 3 .*. .*. .*. .*.	.*. .*. .*. .*.

Задача B. Сжатие по Лемпелю и Зиву

Имя входного файла:	`lz.in`
Имя выходного файла:	`lz.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Многие современные архиваторы, как, например, WinRAR или WinZIP, используют различные разновидности алгоритма Лемпеля-Зива в качестве алгоритма сжатия. Несмотря на то, что разархивирование данных, сжатых по этому алгоритму, обычно весьма несложно, сам процесс сжатия весьма сложен. Поэтому в архиваторах часто часто используются приближенные методы, которые зачастую не обеспечивают максимальной степени сжатия.

Такое положение дел не устраивает Вашего начальника Георгия. Он хочет написать архиватор WinGOR, который должен стать лучшим архиватором. В нем будет использоваться описанная ниже модификация алгоритма LZ77.

Текст разбивается на фрагменты, длина которых не превосходит 4096 символов. Каждый фрагмент после этого сжимается независимо от других. Далее будет описан алгоритм разархивации одного фрагмента (обозначим его t). Основываясь на этом описании, создайте алгоритм, который будет находить минимальный по длине сжатый фрагмент x, соответствующий исходному фрагменту t.

Сжатый фрагмент представляет собой последовательность *обычных символов* и *блоков повторений*. Обычный символ имеет длину 8 битов. При распаковке обычный символ просто копируется в выходной файл. Блок повторения (r, l) состоит из двух частей: *ссылки* и *длины*, каждая из которых имеет длину 12 битов. Ссылка r — это число от 1 до 4095. Когда блок повторения встречается после распаковки $i - 1$ символа текста, символы $t[i - r \ldots i - r + l - 1]$ копируются в выходной файл. Заметим, что r может быть меньше, чем l. В этом случае только что скопированные символы так же копируются в выходной файл.

Для того, чтобы программа-распаковщик могла отличать обычные символы и блоки повторений, к каждому элементу сжатого текста спереди приписывается бит, который равен 0, если этот элемент — обычный символ, и 1 — если это блок повторений.

Например, "`aaabbaaababababababab`" может быть сжат как "`aaabb(5,4)(2,10)`". Сжатый вариант занимает $8 + 8 + 8 + 8 + 8 + 24 + 24 + 7 = 95$ битов вместо 152 битов для исходного варианта (дополнительные 7 битов используются, чтобы различать обычные символы и блоки повторений).

Задан фрагмент текста. Найдите его сжатое представление, которое требует наименьшее количество битов.

Формат входного файла

Входной файл содержит фрагмент текста t. Его длина не превосходит 4096 символов. Он содержит только строчные буквы латинского алфавита.

Формат выходного файла

В первой строке выходного файла выведите длину сжатого текста в битах. Во второй строке выведите сжатый фрагмент. Для обозначения обычных символов используйте сами эти символы, а для обозначения блоков повторения используйте формат "`(r,l)`".

Пример

lz.in	lz.out
aaabbaaababababababab	95 aaabb(5,4)(2,10)

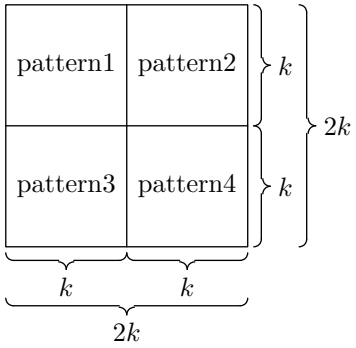
Задача C. Генератор карт

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

Петя пишет генератор случайных карт для новой компьютерной игры *Heroes of Mouse and Keyboard*. Каждая карта в этой игре представляет собой квадратную сетку размером $2^m \times 2^m$, каждый единичный квадрат которой может быть занят либо водой, либо землей.

Генератор, который написал Петя, создает описание карты. Оно имеет следующую структуру. Изначально, есть два шаблона: '1' и '0', обозначающие единичный квадрат, занятый землей или водой, соответственно. Эти шаблоны имеют размер 1. Описание карты представляет собой набор правил вида: "<pattern>=<pattern1>,<pattern2>,<pattern3>,<pattern4>".

Здесь <pattern$i$> — имена уже объявленных шаблонов размера k. После того, как это правило обработано, становится доступным новый шаблон <pattern> размера $2k$. Этот шаблон получается путем расположения шаблонов из правой части правила так, как показано на рисунке.



Петя сгенерировал карту и ему стало интересно, сколько связных компонент земли она содержит. Два единичных квадрата, содержащих землю, принадлежат одной компоненте связности, если у них есть общая сторона. Помогите Пете!

Формат входного файла

Входной файл содержит описание карты. Имена шаблонов состоят только из букв латинского алфавита и цифр. Строчные и прописные буквы различаются, длины имен не превосходят 20 символов. Шаблоны имеют разные имена. Максимальный размер шаблона $2^{16} \times 2^{16}$. Количество шаблонов не превосходит 200.

Сама карта описывается шаблоном, имеющим специальное имя "Map". Описание этого шаблона идет последним в выходном файле. Каждый шаблон, кроме, возможно, "0" или "1", используется при создании карты.

Формат выходного файла

Выведите в выходной файл одно число — количество компонент связности на карте.

Пример

map.in	map.out
A=0,1,1,0 X=1,1,1,1 B=A,A,A,X C=B,B,B,B Map=C,C,C,C	56

Задача D. Мюзикл

Имя входного файла:	`musical.in`
Имя выходного файла:	`musical.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Витя очень любит мюзиклы. Он посетил представления всех мюзиклов мира, некоторые даже по несколько раз! Но сейчас для Вити наступили совсем другие времена. Теперь он — директор собственного мюзикла.

Сейчас он занимается планированием гастролей. Планируется посетить n городов. При этом каждую новую неделю мюзикл будет проходить в новом городе. Представления же будут проходить каждый день.

Витя хочет выбрать порядок посещения городов так, чтобы максимизировать доход. Некоторые города соединены двусторонними дорогами. Несмотря на то, что музыкальное оборудование и сами актеры перемещаются из города в город по воздуху, дороги играют важную роль в том, как музыкальные фанаты посещают представления и обмениваются ифнормацией о мюзикле.

Чтобы оценить доход, Витя собрал информацию о музыкальных фанатах во всех городах. Он разделил их на три категории:

- Обычные музыкальные фанаты посещают мюзиклы раз в неделю. Они посещают только мюзиклы, которые проходят в городе, где они живут. Если же в городе проходит несколько мюзиклов, они выбирают тот, информации о котором у них больше всего. Количество информации зависит от количества соседних городов, уже посещенных мюзиклом. Два города считаются соседними, если они соединены дорогой. Если существует несколько мюзиклов, про которые у них одинаковое количество информации, они выбирают один из них случайно.
- Сумасшедшие музыкальные фанаты ходят на мюзиклы каждый день. Если в городе проходит несколько мюзиклов, они выбирают один из них случайно.
- Наконец, музыкальные маньяки ходят на мюзиклы каждый день, выбирая мюзикл из числа тех, которые идут в городе, в котором они живут, и в соседних городах. Если мюзиклов несколько, выбирается случайный.

Витя уже собрал всю информацию о музыкальных фанатах и о других мюзиклах, которые планируют гастроли в ближайшее время. Помогите Вите выбрать порядок, в котором необходимо посетить города, чтобы максимизировать математическое ожидание количества фанатов, которые посетят его мюзикл.

Формат входного файла

Первая строка входного файла содержит три целых числа: n, m and k — количество городов, дорог и других мюзиклов ($1 \le n \le 10$, $0 \le m$, $0 \le k \le 10$).

Последующие n строк описывают города. Каждый город описывается тремя целыми числами, не превосходящими 10^6, — количеством обычных музыкальных фанатов, сумасшедших музыкальных фанатов и музыкальных маньяков.

Далее идут m строк, описывающих дороги. Каждая дорога описывается двумя числами — номерами городов, которые она соединяет. Никакие два города не соединены более, чем одной дорогой, никакая дорога не соединяет город сам с собой.

Последующие k строк описывают расписание других мюзиклов. Каждый мюзикл описывается n числами. i-ое из этих чисел — номер города, в котором мюзикл гастролирует на i-ой неделе (или 0 — если у мюзикла на этой неделе выходные). Заметим, что, в отличие от Витиного мюзикла, другие мюзиклы могут выступать в одном городе более одного раза.

Формат выходного файла

Выведите в первой строке выходного файла ожидаемое количество фанатов, которые посетят Витин мюзикл. Ваш ответ должен отличаться от правильного не более, чем на 10^{-6}.

Вторая строка должна содержать перестановку n чисел — порядок, в котором города должны быть посещены мюзиклом для максимизации ожидаемого количества посетивших его музыкальных фанатов.

Пример

musical.in	musical.out
3 2 1 100 10 20 20 50 30 10 40 30 1 2 1 3 2 3 1	1670.00000000 1 2 3

В этом примере, на первой неделе ожидается, что мюзикл посетят 555 фанатов — 100 обычных, 10 сумасшедших каждый день (итого, 70), $20/2 = 10$ маньяков из первого города, $30/2 = 15$ маньяков из второго города, 30 маньяков из третьего города в день (итого, 385 в неделю).

На второй неделе, ожидается, что мюзикл посетят $20 + 50 \cdot 7 + (20/2 + 30) \cdot 7 = 650$ фанатов, а на третьей — $10 + 40 \cdot 7 + (20/2 + 30/2) \cdot 7 = 465$. Итого, за три недели: 1670 музыкальных фанатов.

Задача E. Головоломка про ферзей

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

Многие из Вас, наверно, хотя бы раз в жизни играли в шахматы. Поэтому, Вы наверняка знаете, что ферзь — это шахматная фигура, которая может перемещаться по вертикали, горизонтали и диагонали на любое количество полей.

Слава недавно начал заниматься шахматами. На первом занятии учитель рассказал ему, как ходят шахматные фигуры. Слава сразу понял, что ферзь — самая "мощная" шахматная фигура.

Кроме обычной игры в шахматы, Славе сразу полюбились шахматные головоломки. В недавно купленной им книге «Шахматные головоломки для детей от 9 до 99» он обнаружил такую головоломку: "Расставить k ферзей на доске 8 на 8 так, чтобы хотя бы одно свободное поле не было атаковано."

Напишите программу, которая поможет Славе решить эту головоломку.

Поле называется *атакованным*, если хотя бы одна фигура на доске может совершить ход на это поле.

Формат входного файла

Входной файл содержит натуральное число k ($0 \le k \le 64$).

Формат выходного файла

В первой строке выходного файла выведите слово `YES`, если искомая расстановка возможна, и слово `NO` в противном случае. В случае положительного ответа, выведите соответствующую расстановку.

Для этого выведите 8 строк по 8 символов в каждой. Строки соответствуют горизонталям шахматной доски, столбцы — вертикалям. Пустое поле обозначайте символом `.` (точка), поле, на котором стоит ферзь — символом `Q`.

Если существует несколько расстановок ферзей, выведите любую.

Примеры

queens.in	queens.out
1	YES ....Q.... ....
60	NO
5	YES Q......Q ....Q .... Q......Q

Задача F. Кофейни Starbugs

Имя входного файла:	`starbugs.in`
Имя выходного файла:	`starbugs.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Владельцы новой межгалактической сети кофеен *Starbugs* планируют открыть не более, чем k новых кофеен в галактике XXX. В этой галактике находятся n звезд, рядом с которыми находятся обитаемые планеты, i-ая звезда находится в точке (x_i, y_i, z_i). Все координаты целые, заданы в парсеках.

Поскольку межзвездные путешествия — удовольсвтвие не из дешевых, обитатели некоторой звездной системы согласятся посещать кофейню тогда и только тогда, когда она расположена либо в этой звездной системе, либо на расстоянии не более 0.5 парсека от нее. Таким образом, каждая кофейня будет открыта либо рядом со звездой, либо между двумя звезадами, на расстоянии 0.5 парсека от каждой. Люди из звездной системы, в которой есть кофейня или на расстоянии в 0.5 парсека от которой есть ровно одна кофейня, будут посещать только эту кофейню. Если же на расстоянии в 0.5 парсека от звезды есть несколько кофеен, то люди из этой звездной системы будут посещать все эти кофейни.

Менеджеры компании не хотят, чтобы какая-та из кофеен обслуживала слишком мало клиентов. Они собрали информацию о населении p_i каждой из звездных систем. Теперь они хотят открыть новые кофейни так, чтобы жители каждой из звездных систем могли посетить хотя бы одну из кофеен и минимальное среди всех кофеен количество клиентов, обслуживаемых этой кофейней, было максимальным.

Формат входного файла

Первая строка входного файла содержит числа n и k ($1 \le n \le 500$, $1 \le k \le 500$).

Последующие n строк описывают звезды. Каждая строка содержит четыре целых числа — x_i, y_i, z_i и p_i ($-10^4 \le x_i, y_i, z_i \le 10^4$, $1 \le p_i \le 10^8$).

Формат выходного файла

Первая строка входного файла должна содержать два целых числа: l — количество кофеен, которые надо открыть, и максимальное возможное t такое, что каждая кофейня обслуживает хотя бы t клиентов. Последующие l строк должны содержать каждая по три вещественных числа — координаты кофейни.

Если невозможно открыть сеть кофеен, удовлетворяющую всем требованиям, выведите -1 в единственной строке выходного файла.

Пример

starbugs.in	starbugs.out
4 3 0 0 0 9 0 0 1 5 0 1 0 8 0 1 1 3	3 11 0 0 0.5 0 0.5 0 0 1 0.5
2 1 0 0 0 1 1 1 1 1	-1

Задача G. Треугольные страны

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

Эта история происходила на одной плоской планете. С незапамятных времен на ней существовал город N, находящийся в точке x_N, y_N. Кроме этого, в разное время на этой же планете существовали страны, каждая из которых имела форму треугольника.

Теперь перед историками встала серьезная задача — по имеющимся у них данным о треугольных странах определить, в какие страны мог входить город N. Город мог входить в страну, если он находится строго внутри нее.

Формат входного файла

Первая строка входного файла содержит два числа: x_N и y_N — координаты города N. Вторая строка входного файла содержит количество k треугольных стран ($1 \le k \le 1000$). Последующие k строк каждая описывают одну треугольную страну. Описание треугольной страны состоит из шести целых чисел $x_1, y_1, x_2, y_2, x_3, y_3$, где (x_1, y_1), (x_2, y_2), x_3, y_3 — координаты вершин этой страны.

Гарантируется, что все страны имеют ненулевую площадь. Все координаты не превосходят 10000 по абсолютной величине.

Формат выходного файла

В первой строке выходного файла выведите количество стран, в которые мог входить город N. Во второй строке выведите через пробел номера этих стран в возрастающем порядке. Страны нумеруются с единицы в том порядке, в каком они заданы во входном файле.

Примеры

tri.in	tri.out
0 1 2 -2 0 2 0 0 2 -3 0 3 0 0 3	2 1 2
0 2 2 -2 0 2 0 0 2 -3 0 3 0 0 3	1 2

Задача H. Две последовательности

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

Определим последовательности a_n и b_n следующим образом: $a_1 = 2$, $a_2 = 3$, $a_3 = 4$, $a_4 = 7$, $a_n = b_{n-1} + b_{n-3}, n > 4$, b_n — последовательность чисел, не входящих в a_n, записанных в возрастающем порядке.

Таким образом, последовательность a_n будет выглядеть следующим образом: $2, 3, 4, 7, 13, 15, \ldots$, а последовательность b_n — $1, 5, 6, 8, 9, 10, \ldots$.

Ваша задача состоит в том, чтобы найти a_n и b_n.

Формат входного файла

Входной файл содержит целое число n $(1 \le n \le 10^7)$.

Формат выходного файла

В первой строке выходного файла выведите a_n, во второй — b_n.

Примеры

twoseq.in	twoseq.out
4	7 8
10	25 16
6578	19731 9868
10000	29995 15000

Задача I. Недостижимые операторы

Имя входного файла:	`unreachable.in`
Имя выходного файла:	`unreachable.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	64 megabytes

Современные языки программирования, такие, как Java, предъявляют очень строгие требования к корректности исходного кода. Например, каждый оператор программы должен иметь возможность выполниться. Программа, представленная ниже, содержит ошибку — последний оператор никогда не выполняется.

```
int count(int x) {
    if (x >= 10) {
        return 10;
    } else {
        return x + 1;
    }
    return x + 2;
}
```

Компания Карельские Компьютеры собирается начинает разработку нового языка программирования Kava. Он будет содержать небольшое подмножество языка программирования Java. Вы должны создать часть Kava-компилятора, которая будет определять недостижимые операторы. Поскольку Kava является функционально-ориентированным языком, анализировать придется только одну функцию.

Грамматика, описывающая язык программирования Kava, представлена ниже. Здесь $\langle\text{expression}\rangle^*$ означает, что $\langle\text{expression}\rangle$ может повторяться ноль и более раз.

$$
\begin{aligned}
\langle\text{function}\rangle &\longrightarrow \langle\text{id}\rangle\ (\ )\ \{\langle\text{body}\rangle\}\\
\langle\text{body}\rangle &\longrightarrow \varepsilon \mid \langle\text{statement}\rangle\,\langle\text{body}\rangle\\
\langle\text{statement}\rangle &\longrightarrow \langle\text{if-statement}\rangle \mid \langle\text{assign-statement}\rangle \mid \langle\text{return-statement}\rangle \mid \{\langle\text{body}\rangle\}\\
\langle\text{if-statement}\rangle &\longrightarrow if\ (\langle\text{id}\rangle)\ \langle\text{statement}\rangle \mid if\ (\langle\text{id}\rangle)\ \langle\text{statement}\rangle\, else\, \langle\text{statement}\rangle\\
\langle\text{assign-statement}\rangle &\longrightarrow \langle\text{id}\rangle = \langle\text{id}\rangle\,;\\
\langle\text{return-statement}\rangle &\longrightarrow return\,\langle\text{id}\rangle\,;\\
\langle\text{id}\rangle &\longrightarrow \langle\text{char}\rangle\,\langle\text{char}\rangle^*\\
\langle\text{char}\rangle &\longrightarrow A \mid B \mid \ldots \mid Z \mid a \mid b \mid \ldots \mid z
\end{aligned}
$$

Как всегда, в случае неоднозначности `else` соответствует ближайшему к нему оператору `if`.

Оператор называется недостижимым, если при любых результатах вычисления условий в операторах `if` он никогда не выполняется. При этом, не надо делать никаких предположений об отношениях между различными операторами `if` (см. последний пример).

Формат входного файла

Входной файл содержит одну функцию. Гарантируется, что она синтаксически корректна. Размер входного файла не превосходит 20 килобайт. Элементы программы могут быть разделены произвольным числом пробелов. Заглавные и строчные буквы в именах и ключевых словах различаются.

Формат выходного файла

Для каждого недостижимого оператора выведите одну строку, указывающую, в какой строке выходного файла он начался.

Следуйте формату, приведенному в примерах. Если недостижимый оператор является частью другого недостижимого оператора (`if` или блок, окруженный фигурными скобками), его выводить не надо (см. третий пример).

Пример

unreachable.in	unreachable.out

`count() {`
`  if (x) {`
`  return y;`
`  } else {`
`  return z;`
`  }`
`  return a;`
`}`
	`line 7: unreachable statement`

`count() {`
`  if (x) {`
`  return y;`
`  }`
`  return a;`
`}`	

`count() {`
`  return a;`
`  return`
`  a;`
`  if`
`  (x)`
`  a = b;`
`  a = b; c = d;`
`  {`
`  a = b;`
`  c = d;`
`  }`
`}`
	`line 3: unreachable statement`
	`line 5: unreachable statement`
	`line 8: unreachable statement`
	`line 8: unreachable statement`
	`line 9: unreachable statement`

`count() {`
`  if (x)`
`  return a;`
`  if (x)`
`  a = b;`
`  else`
`  return a;`
`  c = d;`
`}`	