Задача A. Максимальное подмножество точек

Имя входного файла:	`colinear.in`
Имя выходного файла:	`colinear.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Институт геометрических открытий (ИГО) решает новую задачу. Задача слишком сложная, поэтому они позвали Васю на помощь. А он, как обычно, немедленно попросил о помощи вас. Вам дано n точек на плоскости. Ваша цель — найти максимальное подмножество этих точек, такое, что никакие три точки в этом подмножестве не лежат на одной прямой.

Формат входного файла

Входной файл состоит не более чем из десяти тестовых наборов. Каждый тестовый набор начинается с целого числа n ($1 \leqslant n \leqslant 30$), затем идут n строк, каждая из которых описывает координаты x_i и y_i соответствующей точки. Все значения координат целые и по модулю не превосходят 10 000. Входной файл заканчивается пустым набором, который состоит из одного числа ноль.

Формат выходного файла

Для каждого тестового набора по указанному в примере (см. ниже) формату сначала выведите количество точек в искомом подмножестве, а в следующей строке — номера этих точек в любом порядке. Точки нумеруются с 1 до n согласно их появлению во входном файле. Вывод для каждого последующего тестового набора отделяется от предыдущего пустой строкой.

Пример

colinear.in

```
1
0 0
3
0 0
1 1
2 2
9
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
0
```

colinear.out

```
Set 1: The largest subset contains 1 points.
  Points indices are 1.

Set 2: The largest subset contains 2 points.
  Points indices are 1,2.

Set 3: The largest subset contains 6 points.
  Points indices are 1,6,8,2,4,9.
```

Задача B. Накрой точки

Имя входного файла:	`cover.in`
Имя выходного файла:	`cover.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сегодня Васю послали в Институт шаров и кругов (ИШаК). Этот институт занимается изучением задач, связанных с шарами и кругами. Там Васе предложили решить следующую задачу.

Дано n точек на плоскости. Требуется покрыть все эти точки в точности k возможно пересекающимися или касающимися кругами одинакового минимально возможного радиуса.

Формат входного файла

Входной файл состоит из одного или более тестовых наборов. Каждый тестовый набор начинается со строки, содержащей два целых числа n и k ($1 \leqslant n \leqslant 15$, $1 \leqslant k \leqslant n$), далее следует n строк, каждая из которых описывает координаты x_i и y_i соответствующей точки. Точки могут совпадать. Все значения координат целые и по модулю не превосходят 1 000. Ввод заканчивается пустым тестовым набором $n = k = 0$. Общее количество точек во всех тестовых наборах входного файла не превосходит 150.

Формат выходного файла

Для каждого тестового набора по указанному в примере (см. ниже) формату сначала выведите значение минимального радиуса. Далее выведите любой способ покрытия исходных точек кругами этого радиуса. Выведенные значения радиуса и координат центров кругов должны не более чем на 10^{-6} отличаться от точных. Мы рекомендуем выводить вещественные числа с максимально возможной точностью. Заметим, что круг нулевого радиуса покрывает одну точку.

Вывод для каждого последующего тестового набора отделяется от предыдущего пустой строкой.

Примеры

cover.in
3 2 0 0 0 1 0 2 3 1 0 0 0 1 1 0 0 0

cover.out

Case 1: The minimal possible radius is 0.5
  circle 1 at (0.0, 0.5)
  circle 2 at (0.0, 2.0)
 	
Case 2: The minimal possible radius is 0.7071067811865476
  circle 1 at (0.5, 0.5)

Задача C. Уимблдонский турнир

Имя входного файла:	`cup.in`
Имя выходного файла:	`cup.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сегодня Институт беспорядка и случайностей (ИБиС), занимающийся теорией вероятностей, поручил Васе написать программу, которая вычисляет вероятности выигрыша Уимблдонского турнира каждым из игроков.

В первом раунде турнира участвует N игроков, N является степенью двойки. Все игроки получают номера от одного до N. В каждом раунде каждый игрок с нечётным номером играет с игроком со следующим чётным номером. Для следующего раунда победитель матча получает номер $\lceil \frac{l}{2} \rceil$, где l — номер игрока в текущем раунде. Проигравшие выбывают. Победитель финального раунда получает титул Чемпиона Турнира.

Формат входного файла

Входной файл состоит из одного или более тестовых наборов. Каждый тестовый набор начинается со строки, содержащей целое N ($2 \leqslant N = 2^k \leqslant 256$). Следующие N строк содержат матрицу вероятностей P; элемент i-й строки и j-го столбца содержит вероятность того, что i-й игрок выиграет у j-го. Так как никакой матч не может закончиться ничьей, сумма p_{ij} и p_{ji} для различных i и j всегда равна 1. Значение p_{ii} не должно приниматься во внимание. Входной файл заканчивается пустым набором, который состоит из одного числа ноль.

Общий размер входного файла не превышает 10^6 символов.

Формат выходного файла

Для каждого турнира по указанному в примере (см. ниже) формату выведите вероятности победы каждого из теннисистов. Выведенные значения вероятностей должны не более чем на 10^{-6} отличаться от точных. Мы рекомендуем выводить вещественные числа с максимально возможной точностью.

Вывод для каждого последующего тестового набора отделяется от предыдущего пустой строкой.

Примеры

cup.in

```
4
1 0 1 1
1 1 0 1
0 1 1 1
0 0 0 1
4
1 0.25 0.25 0.25
0.75 1 0.4 0.25
0.75 0.6 1 0.25
0.75 0.75 0.75 1
0
```

cup.out

```
Tournament #1:
  The probability for player 1 to become the Champion is equal to 0.0
  The probability for player 2 to become the Champion is equal to 0.0
  The probability for player 3 to become the Champion is equal to 1.0
  The probability for player 4 to become the Champion is equal to 0.0

Tournament #2:
  The probability for player 1 to become the Champion is equal to 0.0625
  The probability for player 2 to become the Champion is equal to 0.215625
  The probability for player 3 to become the Champion is equal to 0.159375
  The probability for player 4 to become the Champion is equal to 0.5625
```

Задача D. Разрезание прямоугольника

Имя входного файла:	`cutrect.in`
Имя выходного файла:	`cutrect.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Институт добавления и отрезания (ИДиОт) привлёк Васю к решению новой задачи.

Вам дан прямоугольник размера $m \times n$, вырезанный из листа клетчатой бумаги по линиям сетки. Требуется получить из него прямоугольник размера $s \times t$. Чтобы сделать это, вы можете проводить любой прямолинейный разрез по линиям сетки, но каждая такая операция сопровождается штрафом. Величина штрафа равна отношению площади большего из получающихся при разрезе прямоугольников к площади меньшего из них. В любой момент времени кроме непосредственно разрезания выбранный прямоугольник разрешается поворачивать и переворачивать.

Васе требуется написать программу, которая определит последовательность операций разрезания, позволяющую получить требуемый прямоугольник с минимальным общим штрафом.

Формат входного файла

Входной файл состоит из одного или более тестовых наборов. Каждый набор задан одной строкой, содержащей 4 положительных целых числа m n s t. Ввод заканчивается пустым набором, состоящим из 4-х нулей. Гарантируется, что общая сумма $m + n$ во всех наборах одного входного файла не превышает 800, и во всех наборах второй прямоугольник из первого получить можно.

Формат выходного файла

Для каждого набора в первой строке выведите по приведённому в примере формату значение минимального штрафа с максимально возможной точностью. Затем следует привести пример последовательности операций, с помощью которой можно достичь цели с указанным штрафом. Если решений несколько, выведите любое из них.

Ответы для различных наборов разделяйте пустой строкой.

Примеры

cutrect.in
3 3 1 1 5 5 2 3 0 0 0 0

cutrect.out

Cut 1: The smallest possible total penalty is 4.0
  Cut 3x3 to 2x3 and 1x3, penalty is 2.0
  Cut 3x1 to 1x1 and 2x1, penalty is 2.0
 	
Cut 2: The smallest possible total penalty is 3.0
  Cut 5x5 to 2x5 and 3x5, penalty is 1.5
  Cut 2x5 to 2x3 and 2x2, penalty is 1.5

Задача E. Разрезание прямоугольника-2

Имя входного файла:	`cutrect2.in`
Имя выходного файла:	`cutrect2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

После разрезания прямоугольника из предыдущей задачи Васю попросили разрезать ещё один прямоугольник для Института добавления и отрезания.

Этот прямоугольник размера $m \times n$ также вырезан из листа клетчатой бумаги по линиям сетки. Но теперь некоторые клетки прямоугольника (как минимум одна) объявлены важными. Вася должен разрезать этот прямоугольник прямой линией в заданном направлении так, чтобы отношение общей площади важных клеток в получающихся частях стало равно заданному.

Можете ли вы снова ему помочь?

Формат входного файла

Входной файл содержит не более чем две тысячи тестовых наборов. Каждый тестовый набор начинается со строки, содержащей три числа m, n и R, где m и n целые числа от 1 до 100, а R — это отношение площади левой части к площади правой части. «Левая» и «правая» части определены в соответствии с направлением ненулевого вектора направления разреза. Вектор направления разреза задаётся во второй строке двумя целыми числами dx и dy. Оба этих числа не превосходят 1000 по абсолютному значению. Остальные m строк тестового набора содержат прямоугольную карту: n символов в каждой строке, где '`*`' обозначает важную клетку, а '`.`' обозначает неважную клетку. Входной файл завершается строкой из трёх нулей.

Гарантируется, что R — это вещественное число между 0.001 и 1000, заданное с максимально возможной точностью. Общее число клеток во всех тестовых наборах в одном входном файле не превосходит $2 \cdot 10^6$.

Формат выходного файла

Для каждого тестового набора выведите координаты одной из точек на линии разреза с максимально возможной точностью. Для вывода придерживайтесь формата вывода, показанного в примере ниже. Левый нижний угол прямоугольника имеет координаты $(0, 0)$.

Допускается абсолютная ошибка в определении отношения важных площадей не более 10^{-6} в предположении, что ваш ответ выведен с максимальной возможной точностью.

Примеры

cutrect2.in	cutrect2.out
1 1 1.0 -1 1 * 1 1 7.0 -1 1 * 1 1 7.0 1 -1 * 2 3 9.0 2 -3 *.* .** 0 0 0	Cut rectangle 1 at (0.5, 0.5) Cut rectangle 2 at (1, 0.5) Cut rectangle 3 at (0, 0.5) Cut rectangle 4 at (-0.011237390046160782, 2)

Задача F. Очистка строк

Имя входного файла:	`delete.in`
Имя выходного файла:	`delete.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В Институте красивых строк (ИКС) существует отдел Очистки и зачистки. Теперь перед ними стоит новая задача об очистке строк.

Строки состоят из алфавитно-цифровых символов, регистр которых значим. Допускаются только латинские буквы. Каждый возможный символ имеет ассоциированное с ним число. Процесс очистки строки состоит из нескольких итераций.

Каждая итерация начинается с удаления первого символа в строке. Затем указатель текущего положения перемещается на k_c символов вперёд, где k_c — положительное целое число, ассоциированное с удалённым символом. Символ в новом положении указателя снова удаляется, и процесс повторяется до тех пор, пока указатель не выйдет за пределы строки.

Рассмотрим следующий пример. Пусть символу '`a`' соответствует число 1, символу '`b`' — 2, а символу '`c`' — 3. На первой итерации очистки строки "`abacaba`" удаляется символ '`a`', затем указатель перемещается на символ '`b`', удаляет его, затем перемещается на '`c`', удаляет его и перемещается на последний '`a`', удаление которого завершает итерацию. Оставшаяся строка равна "`aab`". Если мы теперь запустим ещё одну итерацию, эта строка будет очищена полностью.

Задача Васи состоит в том, чтобы найти число итераций, необходимых для очистки данной строки. Кроме того, он должен найти порядок удалений. Можете ли вы помочь ему?

Формат входного файла

Входной файл состоит не более чем из 1000 тестовых наборов. Каждый тестовый набор начинается с числа допустимых символов n ($1 \leqslant n \leqslant 62$), за которым следуют n строк, каждая из которых содержит символ c и число k_c, ассоциированное с ним ($1 \leqslant k_c \leqslant 10^9$). Символы перечисляются в произвольном порядке и не повторяются. Последняя строка в каждом тестовом примере — это непустая строка для обработки, которая состоит только из перечисленных выше символов. Ввод заканчивается со значением $n = 0$, этот пустой тестовый пример не должен обрабатываться.

Общий размер всех строк для обработки во входных данных не превосходит 10^6 символов.

Формат выходного файла

Для каждого тестового набора выведите число итераций, необходимое для полной очистки строки, затем выведите строку, описывающую порядок удалений. Придерживайтесь формата вывода, показанного в примерах. Результаты для тестовых наборов должны быть разделены одной пустой строкой.

Примеры

delete.in	delete.out
3 a 1 b 2 c 3 abacaba 3 9 2 c 3 a 1 aaa 2 a 1 A 3 AAaA 6 a 3 b 2 c 3 1 2 2 3 3 2 abc123 0	String 1: 2 iterations required. Deleted in order: abca,aab String 2: 1 iterations required. Deleted in order: aaa String 3: 3 iterations required. Deleted in order: AA,A,a String 4: 3 iterations required. Deleted in order: a13,b2,c

Задача G. Вложение графа

Имя входного файла:	`embedding.in`
Имя выходного файла:	`embedding.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Институт красивых строк (ИКС) получил просьбу о помощи от Матрично-графового университета (МГУ). Темой исследования является вложение произвольных неориентированных графов в бесконечные бинарные деревья. Более подробно задача описана ниже.

Пусть $\mathcal{G} = (V, E)$ — граф, вершины которого занумерованы от 1 до n включительно. Пусть $\mathcal{T}$ — бесконечное корневое бинарное дерево, узлы которого помечены в соответствии со следующими правилами:

- Корень помечен ε (пустая строка).
- Для каждого $t \in \mathcal{T}$ его левый сын помечен $N(t) \cdot$ '`l`', а правый сын помечен $N(t) \cdot$ '`r`', где $N(t)$ — это метка вершины t, а операция '$\cdot$' — это конкатенация.

Таким образом, сыновья корневого узла помечены '`l`' и '`r`', их четыре сына помечены '`ll`', '`lr`', '`rl`' и '`rr`', и т. д.

Вложение графа $\mathcal{G}_1 = (V_1, E_1)$ в другой граф $\mathcal{G}_2 = (V_2, E_2)$ — это отображение $f : V_1 \to V_2$ такое, что выполняются следующие условия:

- Различные вершины $\mathcal{G}_1$ отображаются в различные вершины $\mathcal{G}_2$;
- Для каждого ребра $e_1 \in E_1$, соединяющего u и v, существует ребро $e_2 \in E_2$, соединяющее $f(u)$ и $f(v)$.

Другими словами, мы инъективно отображаем V_1 в подмножество V_2 так, что E_1 соответствует некоторому подмножеству E_2.

Будем обозначать вложение f графа $\mathcal{G}$ в дерево $\mathcal{T}$ как $(f(v_1), f(v_2), \ldots f(v_n))$ — список строк, которые соответствуют узлам в которые отображены вершины $1, 2, \ldots, n$.

Например, если $V = \{1, 2, 3\}$, $E = \{(1, 2), (1, 3)\}$, и мы отображаем первую вершину в корень, вторую вершину в правый сын корня и третью вершину в левый сын корня, такое отображение будет вложением и записываться ('', '`r`', '`l`'). С другой стороны, если мы отобразим вторую вершину на корень, а остальные две вершины на сыновья корня, это не будет допустимым вложением, так как 1 и 3 должны быть соединены ребром, а сыновья корня дерева не соединены.

По данному графу $\mathcal{G}$ и целому числу k найдите k-е в лексикографическом порядке вложение графа $\mathcal{G}$ в дерево $\mathcal{T}$. k отсчитывается от нуля.

Формат входного файла

Входной файл состоит из не более чем десяти тестовых наборов. Каждый тестовый набор начинается со строки, содержащей три целых числа n, m и k разделённых одним пробелом ($1 \leqslant n \leqslant 10$, $0 \leqslant k \leqslant 1\,000\,000$). Следующие m строк описывают ребра, каждая строка содержит два целых числа u_i и v_i, разделённых одним пробелом — конечные точки ребра ($1 \leqslant u_i, v_i \leqslant n$). Граф не содержит циклов и содержит не более чем одно ребро между любой парой вершин. Строка из трёх нулей завершает ввод, этот тестовый набор не должен обрабатываться. Сумма всех k во входном файле не превышает $1\,000\,000$.

Формат выходного файла

Для каждого тестового набора выведите запись k-го (считая от нуля) вложения графа $\mathcal{G}$ в дерево $\mathcal{T}$ в лексикографическом порядке или сообщите, что такого вложения не существует. Придерживайтесь спецификации вывода, показанной на примере ниже.

Примеры

embedding.in

```
3 2 0
1 2
1 3
3 2 1
1 2
1 3
3 2 4
2 1
2 3
3 2 5
2 1
2 3
3 3 1
1 2
2 3
3 1
5 4 20
3 1
3 2
3 4
3 5
2 0 1
0 0 0
```

embedding.out

```
Case 1: The embedding number 0 is (‘’, ‘l’, ‘r’).
Case 2: The embedding number 1 is (‘’, ‘r’, ‘l’).
Case 3: The embedding number 4 is (‘l’, ‘’, ‘r’).
Case 4: The embedding number 5 is (‘l’, ‘ll’, ‘lll’).
Case 5: There is no such embedding.
Case 6: There is no such embedding.
Case 7: The embedding number 1 is (‘’, ‘ll’).
```

Задача N. Жизнь

Имя входного файла:	`life.in`
Имя выходного файла:	`life.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Институт закономерных биологических аномалий (ИЗБА) открыл очень странную колонию микроорганизмов. Микроорганизмы живут на бесконечной двухмерной квадратной сетке из клеток. Каждая клетка взаимодействует со своими восемью соседями по вертикали, горизонтали или диагоналям. После долгих наблюдений один известный учёный вывел законы поведения популяции:

1. Каждый микроорганизм с двумя или тремя соседями выживает.
2. Каждый микроорганизм с четырьмя или более соседями умирает от перенаселения.
3. Каждый микроорганизм с одним или менее соседом умирает от одиночества.
4. Каждая пустая клетка ровно с тремя живыми соседями оживает.

Следующее поколение получается применением вышеперечисленных правил одновременно к каждой клетке в текущей популяции. Рождения и смерти происходят одновременно.

Несмотря на то, что сформулированные законы очень просты, колония микроорганизмов развивается совершенно непредсказуемо. Васю попросили написать программу, которая определит, сколько микроорганизмов окажутся живыми в K-м поколении колонии, начиная с заданного (нулевого) поколения.

Перед тем, как начать писать программу, Вася нарисовал несколько примеров развития колонии на листе бумаги. Очень скоро он обнаружил очень странную колонию:

```
.o..       ....       ....       ....       ....
..o.  ==>  o.o.  ==>  ..o.  ==>  .o..  ==>  ..o.
ooo.       .oo.       o.o.       ..oo       ...o
....       .o..       .oo.       .oo.       .ooo
```

Обнаружив это, Вася вдруг понял, что не знает, как промоделировать процесс эффективно. Помогите Васе написать эффективную программу.

Формат входного файла

Первая строка входного файла содержит одно целое число N ($1 \leqslant N \leqslant 16$) — число живых микроорганизмов в нулевом поколении. Далее следует N строк, каждая из которых описывает один микроорганизм по его целым координатам x_i и y_i ($0 \leqslant x_i, y_i < 4$). Координаты всех N микроорганизмов различны. Оставшаяся часть входного файла содержит один или более тестовых наборов. Каждый тестовый набор задаётся в одной строке одним целым числом K ($1 \leqslant K \leqslant 30\,000$). Входные данные завершаются пустым тестовым набором, состоящим из одного нуля. Во входном файле находится не более чем сто тестовых наборов.

Формат выходного файла

Для каждого тестового набора ваша программа должна вывести размер K-го поколения популяции. Придерживайтесь формата вывода результата, показанного на примере ниже.

Пример

life.in
4 0 0 1 0 1 1 2 0 1 2 4 6 8 0

life.out
Case 1: move 1, population size equals 7. Case 2: move 2, population size equals 6. Case 3: move 4, population size equals 8. Case 4: move 6, population size equals 12. Case 5: move 8, population size equals 20.

Задача I. Простые числа в последовательности

Имя входного файла:	`primes.in`
Имя выходного файла:	`primes.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

ИПроЧиЕ (Институт простых чисел имени Ершова) поставил перед Васей задачу поиска простых чисел в хитрых последовательностях:

Хитрая последовательность начинается некоторым целым положительным числом a_1. Последующие члены хитрой последовательности вычисляются по формуле $a_{i+1} = d(a_i) + i$, где $d(x)$ — количество различных целых положительных делителей числа x. Например, если хитрая последовательность начинается с $a_1 = 2^{32\,582\,657} - 1$, то $a_2 = 3$, $a_3 = 4$ и $a_4 = 6$.

Определим (l, r)-*окно* некоторой последовательности как её подпоследовательность, начинающуюся с l-го элемента и заканчивающуюся r-м элементом.

Васина задача заключается в том, чтобы по заданным l и r определить наибольшее количество элементов (l, r)-окна хитрой последовательности, являющихся простыми числами. Например, указанная последовательность является примером «оптимальной» хитрой последовательности для $(1, 4)$-окна, порождающей 2 простых числа.

Так как хитрых последовательностей бесконечно много, то у Васи не хватит времени разобрать все такие последовательности, и он просит Вас помочь ему с решением задачи.

Формат входного файла

Входной файл состоит из не более чем 1000 тестовых наборов. Каждый набор представляет собой одну строку, в которой записаны два целых числа l и r, при этом $(1 \leqslant l < r \leqslant 10^6)$. Завершает входной файл набор, состоящий из двух нулей, который обрабатывать не нужно. При этом общая сумма всех $l + r$ для всех наборов в одном входном файле не превышает 10^7.

Формат выходного файла

Выведите максимальное возможное количество простых чисел для (l, r)-окна хитрых последовательностей, придерживаясь формата вывода, указанного в примере.

Примеры

primes.in
1 4 6 7 10 20 0 0

primes.out
Window 1: There can be up to 2 primes. Window 2: There can be up to 1 primes. Window 3: There can be up to 4 primes.

Задача J. Запись подмножеств

Имя входного файла:	`subsets.in`
Имя выходного файла:	`subsets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

ИКС (Институт красивых строк) совместно с ИМП (Институт множеств и подмножеств) исследуют различные способы записи подмножеств.

Пусть элементы множества мощности $n \leqslant 30$ занумерованы числами от 1 до n включительно. Существуют две наиболее часто используемых системы записи этих подмножеств. Первая — перечисление в порядке возрастания номеров всех элементов подмножества, тогда получается список длины, равной мощности подмножества, составленный из различных чисел. Вторая — построение строки длиной n, в которой i-й с начала символ равен 1, если соответствующий элемент присутствует в подмножестве, и 0 в противном случае.

После того, как мы выбрали систему записи, естественно отсортировать подмножества лексикографически в порядке их записи: как список чисел (в первом случае) и как строки (во втором). Таким образом, мы получим различное упорядочение для различных систем записи. Например, ниже даны упорядоченные обозначения 8 подмножеств для $n = 3$ в обеих системах записи.

1		000
2	1	001
3	1 2	010
4	1 2 3	011
5	1 3	100
6	2	101
7	2 3	110
8	3	111

Ваша задача — по заданному обозначению в одной из систем записи найти обозначение в другой системе записи, имеющее тот же номер в соответствующем упорядочении.

Формат входного файла

Первая строка ввода содержит T — количество тестовых наборов. Каждая из следующих T строк содержит запись подмножества в одной из двух систем, перечисленных выше. Если обозначение может быть понято неоднозначно, выбирайте тот вариант, для которого исходное множество содержит наименьшее возможное количество элементов. Если и при этом условии неоднозначность сохраняется, выбирайте любую интерпретацию.

Формат выходного файла

Для каждого тестового набора, представляющего собой подмножество, записанное в системе записи X, выведите в системе записи Y, отличной от X, подмножество, которое, будучи записано в системе Y, при лексикографическом упорядочении имеет тот же номер, что и заданное подмножество в системе X. При выводе придерживайтесь указанного в примере формата.

Примеры

subsets.in	subsets.out
5 100 2 3 5 11 0000001010	Case 1: [1 3] Case 2: [10101] Case 3: [] Case 4: [2] Case 5: [1 2 3 4 5 6 7 8 9 10]

Задача K. Трёхцветные треугольники

Имя входного файла:	`tricolour.in`
Имя выходного файла:	`tricolour.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

ИКОТА (Институт канонических оснований трёхцветной архитектуры) приступил к изучению раскрашенных в три цвета треугольников. Первая задача, с которой столкнулись в институте, была следующей. Дан неориентированный граф, в котором каждое ребро раскрашено в один из трёх цветов: красный, зелёный и синий. Требуется найти количество треугольников в данном графе, которые имели бы рёбра всех трёх цветов (такие треугольники называются *трёхцветными*).

Так как специалисты из ИКОТА не занимались ранее задачами на графах, они предложили Вам решить эту задачу.

Формат входного файла

Входной файл содержит один или несколько тестовых наборов. Каждый набор начинается с двух целых чисел n и m ($2 \leqslant n \leqslant 100$, $1 \leqslant m \leqslant n \cdot (n-1)/2$). Далее идут m строк, каждая из которых описывает одно ребро в виде x y c, где x и y — номера вершин, которые данное ребро соединяет, а c — один из следующих символов: '`R`' для красного цвета, '`G`' для зелёного, '`B`' для синего. Граф не содержит петель и кратных рёбер. Общая сумма всех n в одном входном файле не превышает 1000.

Входной файл завершается тестовым набором с $n = m = 0$, который обрабатывать не надо.

Формат выходного файла

Для каждого тестового набора выведите количество трёхцветных треугольников, придерживаясь формата вывода, приведённого в примере.

Примеры

tricolour.in
3 2 1 2 R 2 3 R 3 3 1 2 R 2 3 G 1 3 B 0 0
tricolour.out
Graph 1: There are 0 tri-colour triangles. Graph 2: There are 1 tri-colour triangles.