Problem D. Decomposable Single Word Languages

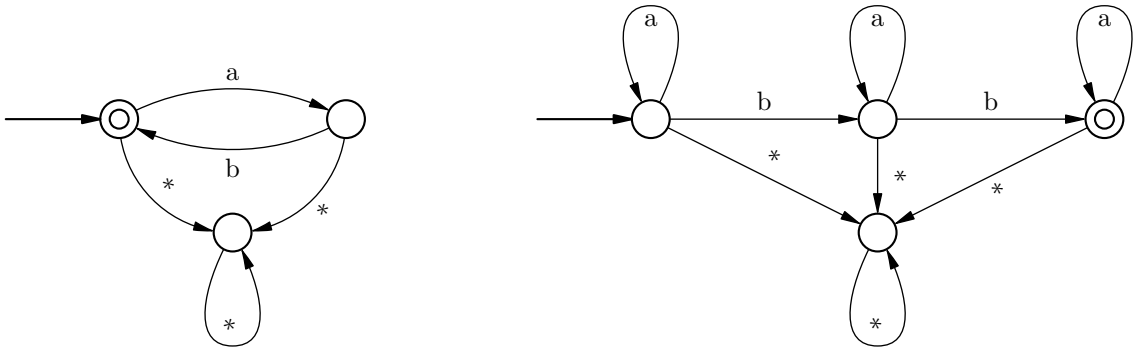
Input file:	`decomposable.in`
Output file:	`decomposable.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Детерминированный конечный автомат (DFA) — это упорядоченное множество $\langle \Sigma, U, S, T, \varphi \rangle$, где Σ — конечное множество, называемое *входным алфавитом*, в этой задаче $\Sigma = \{\text{a}, \text{b}, \ldots, \text{z}\}$, U — конечное множество *состояний*, $S \in U$ — *начальное состояние*, $T \subset U$ — множество *заключительных состояний* и $\varphi : U \times \Sigma \to U$ — *функции переходов*.

На вход автомату подаётся строка α над алфавитом Σ. Изначально автомат находится в состоянии s. На каждом шагу автомат считывает первый символ входной строки c и меняет своё состояние на $\varphi(u, c)$, где u — текущее состояние. После этого первый символ входной строки удаляется и действия повторяются. Если в момент, когда входная строка станет пустой, автомат окажется в одном из заключительных состояний, он принимает строку α, в противном случае он не принимает её. Множество всех слов, принимаемых автоматом A, обозначается как $L(A)$.

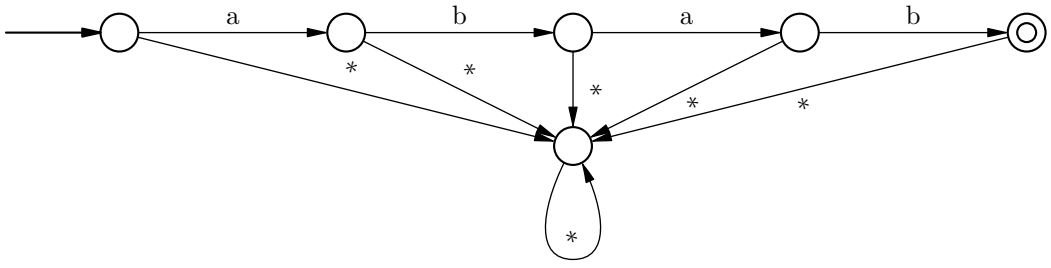
Конечный автомат можно представить как ориентированный граф, в котором вершины соответствуют его состояниям, а переходы — рёбрам, помеченным символами. Заключительные состояния обозначены как вершины, взятые в двойной кружок, начальное состояние помечено стрелкой. На иллюстрации снизу слева показан конечный автомат для языка $(ab)^*$, состоящего из повторяющегося некоторое (возможно, нулевое) количество раз слова “ab”. На иллюстрации снизу справа показан автомат для языка $a^*ba^*ba^*$, состоящего из “a” и “b” и содержащего две “b”.

Для ясности, рёбра, помеченные “$*$” обозначают все переходы, которые явно не заданы.



Назовём множество слов X *регулярным языком*, если оно совпадает с $L(A)$ для некоторого DFA A.*Индекс* регулярного языка X, обозначаемый как $ind(X)$ — это наименьшее количество состояний такого конечного автомата A, что $L(A) = X$. Например, два автомата, изображённых на иллюстрациях, являются минимальными автоматами для заданных языков, так что $ind((ab)^*) = 3$ and $ind(a^*ba^*ba^*) = 4$.

Хорошо известно, что если X_1 и X_2 — два регулярных языка, то их пересечение $X_1 \cap X_2$ тоже является регулярным языком. Например, пересечением двух описанных выше языков является язык $Y = (ab)^* \cap (a^*ba^*ba^*) = \{abab\}$, содержащий одно слово “abab”. Очевидно, что этот язык является регулярным; автомат для него показан на иллюстрации ниже.



Легко увидеть, что если W — однословный язык $W = \{w\}$, а длина слова w равна n, индекс W равен $n + 2$.

Регулярный язык X называется раскладываемым, если X может быть представлен как пересечение двух регулярных языков $X = X_1 \cap X_2$ и $ind(X) > ind(X_1)$ and $ind(X) > ind(X_2)$. Например, однословный язык $Y = \{\text{abab}\}$ является раскладываемым.

По заданному слову w, имеющему длину n, найдите, является ли однословный язык $W = \{w\}$ раскладываемым, и, если является, найдите два автомата A_1 и A_2 такие, что количество состояний как в A_1, так и в A_2 меньше, чем $n + 2$ и $W = L(A_1) \cap L(A_2)$.

Input

Входной файл состоит из нескольких тестовых примеров.

Каждый тестовый пример состоит из слова w, записанного в отдельной строке; w состоит из строчных латинских букв, длина слова w находится между 1 и 50 включительно.

Вхоодной файл содежит не более 100 тестов.

Output

Для каждого тестового примера выведите «YES», если соответствующий однословный язык является раскладываемым, в противном случае выведите «NO».

Если язык является раскладываемым, далее выведите описания двух DFA. Описание каждого автомата должно начинаться с целого числа k — количества состояний, $1 \le k \le n + 1$, где n — длина входного слова. Пусть состояния занумерованы от 1 до k, при этом начальному состоянию соответствует 1; Вам надо вывести t — количество заключительных состояний, $1 \le t \le k$, за которым следуют t целых чисел от 1 до k — номера заключительных состояний. Последующие k строк должны содержать по 26 чисел каждая: для состояния u выведите $\varphi(u, \text{a})$, $\varphi(u, \text{b})$, $\ldots$, $\varphi(u, \text{z})$.

Examples

decomposable.in
abab a
decomposable.out
YES 3 1 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 1 3 2 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 NO

Problem E. Elegant Square

Input file:	`elegant.in`
Output file:	`elegant.out`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Многие знают про магические квадраты — квадраты, содержащие различные целые числа, суммы в строках и в столбцах которых равны. Недавно Ева прочитала про магические квадраты и придумала собственную их версию — *элегантные квадраты*.

Она называет квадрат $n \times n$, составленный из целых чисел, элегантным, если он удовлетворяет следующим условиям:

- Клетки квадрата содержат попарно различные целые положительные числа.
- Все записанные в квадрате числа свободны от квадратов (то есть ни одно из них не делится на t^2 ни для какого $t > 1$).
- Произведения чисел в каждой строке и в каждом столбце совпадают.

Приведём пример элегантного квадрата 3×3.

1	21	10
6	5	7
35	2	3

Все числа положительны, попарно различны и свободны от квадратов, произведения чисел в каждом строке и в каждом столбце равны 210.

Помогите Еве построить элегантный квадрат $n \times n$, состоящий из чисел, не превышающих 10^{18}. Гарантируется, что для ограничений задачи такой квадрат всегда существует.

Input

Входной файл содержит одно целое число n ($3 \le n \le 30$).

Output

Выведите $n \times n$ целых чисел: найденный элегантный квадрат. Выведенные числа не должны превышать 10^{18}.

Examples

elegant.in	elegant.out
3	1 21 10 6 5 7 35 2 3

Problem F. Four Colors

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Это интерактивная задача.

Фред и Фиона устали играть в крестики-нолики на уроках и придумали новую игру. Игровым полем является связный неориентированный граф без циклов, также известный как дерево. Игроки делают ходы по очереди, Фред ходит первым. В игре используются ручки четырёх цветов: красного, зелёного, синего и жёлтого; обозначим эти цвета последовательными целыми числами от 1 до 4.

Первоначально ни одна вершина дерева не окрашена. Игрок на своём ходу выбирает некоторую неокрашенную вершину и окрашивает её в один из четырёх цветов таким образом, что никакие две вершины, соединённые ребром, не являются одноцветными.

Когда очередной игрок не может сделать хода, так как все вершины уже окрашены или же не существует вершины, которую можно окрасить без нарушения правил, игра завершается. Победитель определяется следующим образом: если все вершины окрашены, выигрывает Фред; в противном случае выигрывает Фиона.

Фиона заметила, что Фред постоянно выигрывает; подумав, она поняла, что существует выигрышная стратегия для Фреда. Можете ли Вы найти эту стратегию?

Interaction Protocol

Ваша программа будет играть против программы, написанной жюри, используя стандартные потоки ввода-вывода.

Сначала ваша программа должна считать описание дерева. Дерево задано количеством вершин n ($2 \le n \le 1000$), за которым идёт $n - 1$ пара вершин u_i, v_i, задающая i-е ребро дерева (вершины пронумерованы последовательными целыми числами от 1 до n).

Процесс взаимодействия с программой жюри происходит следующим образом. Ваша программа делает первый ход. Ход состоит в выводе номера вершины u, которая на момент хода не окрашена и цвета, в который вершину надо окрасить, в стандартный поток вывода. Затем программа должна считать ход программы жюри в том же формате, сделать свой ход и так далее, пока игра не закончится.

Когда игра закончилась победой Вашей программы (как в случае, когда последняя вершина была окрашена Вашей программой, так и в случае, когда это был вынужден сделать оппонент), вместо хода оппонента она получит из стандартного потока ввода "-1 -1". После получения -1 -1 (и только после него; даже если Вы вычислили, что выиграли раньше или что сделали последний ход, это не должно вести к завершению работы программы) Ваша программа должна завершить работу.

Examples

standard input	standard output
8 1 2 1 3 2 4 2 5 3 6 3 7 3 8 6 2 8 3 4 2 -1 -1	 1 1 7 2 3 4 2 3

Note

Вход форматирован таким образом, чтобы было видно, какой вывод поступает на соответствующий вход. В реальном взаимодействии пустых строк не будет.

В примере программа жюри делает последний ход (“5 1”, “5 2” или “5 4”), но не сообщает его Вашей программе, так как после этого игра завершается победой Вашей программы.

Problem G. Greater Number Wins

Input file:	greater.in
Output file:	greater.out
Time limit:	2 seconds
Memory limit:	512 megabytes

Георгий и Гордон играют в игру, называемую "Большее число выигрывает".

Игра проходит следующим образом. У каждого игрока есть последовательность из d ячеек. В начале игры все ячейки пусты. В данной задаче рассматриваются две разновидности этой игры.

В первой разновидности участники делают ходы по очереди, Георгий ходит первым. Каждый ход участник бросает игральную кость, которая генерирует случайную b-ичную цифру от 0 до $b-1$, все цифры генерируются с равной вероятностью. После этого участник помещает цифру в одну из свободных ячеек.

После того, как каждый игрок сделал d ходов, игра завершается. Победителем объявляется тот участник, в ячейках которого при чтении слева направо записано наибольшее b-ичное число. Если числа совпадают, игра считается закончившейся вничью.

Второй вариант игры такой же, но сначала Георгий делает d ходов, затем Гордон делает d ходов.

По заданным d и b найдите вероятность выигрыша Георгия в первом и во втором вариантах игры соответственно. Стратегия каждого из участников направлена на то, чтобы максимизировать вероятность своего выигрыша.

Input

Входной файл содержит несколько тестовых примеров.

Каждый тестовый пример содержит два целых числа d и b, заданных в одной строке ($1 \le d \le 10$, $2 \le b \le 10$, $(b+1)^d \le 3000$).

Входной файл завершается строкой с $d = b = 0$.

Output

Для каждого тестового примера выведите два числа в одной строке: вероятность выигрыша Георгия в первой и второй разновидностях игры соответственно с абсолютной точностью не хуже 10^{-6}.

Examples

greater.in	greater.out
1 2 2 2 0 0	0.25 0.25 0.3125 0.3125

В примере в обоих вариантах игры оба игрока должны придерживаться следующей стратегии: Если выпадает единица, её надо поместить в самую левую свободную ячейку, если выпадает 0, его надо поместить в самую правую свободную ячейку.

Если есть только одна ячейка, Георгий выиграет только в случае, если он выбросит 1, а Гордон — 0, вероятность такого исхода равна 1/4. Если есть две ячейки, то к победе Георгия ведут 5 исходов из 16: 11 против 10, 01, или 00; 10 против 00 и 01 против 00.

Problem I. Isomorphism

Input file:	`isomorphism.in`
Output file:	`isomorphism.out`
Time limit:	2 seconds
Memory limit:	512 megabytes

Задача про изоморфизм графов является одной из важных задач в Computer Science. Неизвестно, существует ли полиномиальный алгоритм для этой задачи, также не доказано, что задача не является NP-полной.

Два неориентированных графа G и H называются изоморфными, если у них количество вершин совпадает и существует биекция $\varphi : VG \to VH$ такая, что ребро uv в G существует тогда и только тогда, когда существует ребро $\varphi(u)\varphi(v)$ в H.

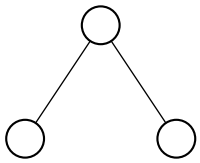
Некоторые характеристики графа являются инвариантными относительно изоморфизма. Одним из *степенным профилем графа.*

Степенью $\deg(u)$ вершины u называется количество вершин, не совпадающих с ней и соединённых с u рёбрами. Рассмотрим связный неориентированный граф G с n вершинами. Для каждой вершины u найдём множества вершин $V_{u,0}, V_{u,1}, \ldots, V_{u,n-1}$ на расстояниях $0, 1, \ldots, n-1$ от u (некоторые из этих множеств могут быть пустыми).

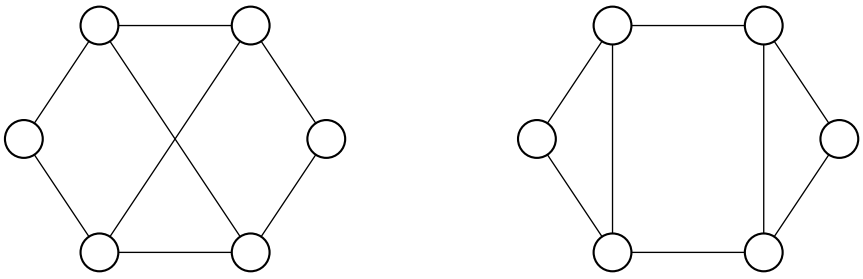
Для каждого такого множества найдём мультимножество $D_{u,i}$ степеней вершин из $V_{u,i}$. Список этих мультимножеств $D_u = [D_{u,0}, D_{u,1}, \ldots, D_{u,n-1}]$ назовём *степенным профилем* вершины u. Мультимножество степенных профилей всех вершин графа назовём степенным профилем графа.

Например, граф, изображённый ниже, имеет степенной профиль

$\big\{[\{1\}, \{2\}, \{1\}], [\{2\}, \{1, 1\}, \varnothing], [\{1\}, \{2\}, \{1\}]\big\}$.



Очевидно, что степенной профиль является инвариантом относительно изоморфизма. Тем не менее, существуют графы, которые имеют тот же самый степенной профиль, но не являются изоморфными. Примеры двух таких графов приведены на иллюстрации внизу. Вершины степени 2 для обоих графов имеют степенные профили $[\{2\}, \{3, 3\}, \{3, 3\}, \{2\}, \varnothing, \varnothing]$, вершины степени 3 имеют степенные профили $[\{3\}, \{2, 3, 3\}, \{2, 3\}, \varnothing, \varnothing, \varnothing]$, но графы не являются изоморфными.



Заметим, что если различие степенных профилей показывает, что графы не изоморфны, то совпадение степенных профилей не даёт простого пути поиска изоморфизма даже в случае, когда он существует, так как может оказаться сложно построить соответствие между вершинами с одинаковым степенным профилем. Однако существует класс графов, для которых степенной профиль даёт возможность простой проверки на изоморфизм. Это графы, в которых все вершины имеют различный степенной профиль. Назовём эти графы *различимыми по степени.*

Даже различимые по степени графы могут иметь совпадающие степенные профили, но не быть изоморфными. Ваша задача — по заданному n найти два неизоморфных связных графа с n вершинами, различимых по степени и имеющих один и тот же степенной профиль.

Input

Входной файл содержит одно целое число n ($3 \le n \le 100$).

Output

Если не существует двух неизоморфных различимых по степени графов с n вершинами, имеющих один и тот же степенной профиль, выведите "NO" в первой строке выходного файла. Иначе выведите "YES", а в следующих строках — описание графа.

Каждое описание должно начинаться числом m — количетсвом рёбер, за которым следует m пар целых чисел: номера вершин, соединённых рёбрами. При этом петель и кратных рёбер быть не должно.

Вершины каждого графа должны быть пронумерованы последовательными целыми числами от 1 до n так, что i-е вершины каждого графа имеют одинаковый степенной профиль. Никакие две вершины одного и того же графа не могут иметь совпадающий степенной профиль.

Note

Во втором примере показаны два неизоморфных графа с одинаковым степенным профилем, которые не являются различимыми по степени. Этот ответ приведён только для иллюстрации выходного формата; соответствующий вывод для $n = 6$ в качестве решения принят не будет.

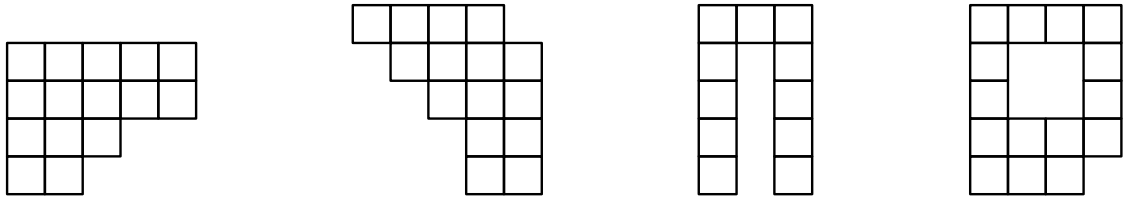
Examples

isomorphism.in	isomorphism.out
3	NO
6	YES 8 1 2 1 6 2 3 2 5 3 4 3 6 4 5 5 6 8 1 2 1 6 2 3 2 6 3 4 3 5 4 5 5 6 *Данный ответ неверен!*

Problem J. Jinxiety of a Polyomino

Input file:	`jinxiety.in`
Output file:	`jinxiety.out`
Time limit:	2 seconds
Memory limit:	512 megabytes

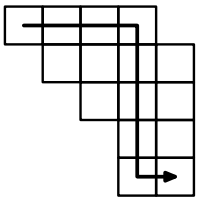
Полимино — это связный по стороне набор единичных квадратов на клетчатом поле. На иллюстрации снизу изображены 4 примера полимино:



(a) (b) (c) (d)

Полимино называется выпуклым, если его пересечение с любой горизонтальной или вертикальной линией является отрезком. На иллюстрации выше полимино (a) и (b) — выпуклые, а (c) и (d) — нет.

Два единичных квадрата называются смежными, если они имеют общую сторону. Легко заметить, что для любых двух квадратов выпуклого полимино возможно добраться от одного составляющего его квадарата до любого другого перемещаясь по смежным квадратам и двигаясь в двух направлениях. Пример снизу иллюстрирует пример подобного пути для полимино (b).



Для выпуклого полимино P определим его *изломанность* $J(P)$ как наименьшее k такое, что существует путь между любыми двумя квадратами, использующий только два направления и содержащий не более k поворотов. Например, изломанность полимино a равна 1, а изломанность полимино (b) равна 2.

Вам задано выпуклое полимино. Определите его изломанность.

Input

Входной файл содержит несколько тестовых примеров.

Каждый тестовый пример содержит два целых числа h и w — количество строк и столбцов, соответственно, в описании полимино ($1 \le h, w \le 2000$).

Последующие h строк содержат w символов каждая и задают полимино. Каждый символ является или точкой "`.`", обозначающей пустой квадрат, или "`#`", обозначающим квадрат, принадлежащий полимино. Гарантируется, что заданная фигура является выпуклым полимино.

Входной файл завершается строкой с $h = w = 0$, обрабатывать которую не следует.

Общее количество символов во всех описаниях полимино не превосходит $4 \cdot 10^6$, а общее количество тестов не превосходит 40 000.

Output

Для каждого тестового примера выведите одно целое число — изломанность полимино, заданном в соответствующем примере.

Examples

jinxiety.in	jinxiety.out
4 5 ##### ##### ###.. ##... 5 5 ####. .#### ..### ...## ...## 0 0	1 2

Problem K. Kill The PSU

Input file:	`killthepsu.in`
Output file:	`killthepsu.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Вам поручена разработка системы контроля стабильности работы устройств и их драйверов. Система должна выдавать подсказки пользователю в зависимости от типа устройства, с которым произошёл сбой.

Устройства с точки зрения этой системы делятся на следующие три категории (каждое устройство принадлежит только к одной из них).

- Категория 1: Внешняя периферия. Проблемы этих устройств заключаются в том, что время от времени они переходят в режим экономии питания (засыпают). В случае, если это устройство встретилось в логе, Вы должны рекомендовать пользователю разбудить (wake) его.
- Категория 2: Внутренние устройства с нестабильными драйверами. При появлении проблем с такими устройствами требуется загрузить (load) драйвер, если он не был загружен, или выгрузить в противном случае. Так как драйверы являются экспериментальными, то в момент начальной загрузки система их не загружает.
- Категория 3: Основные устройства. Проблемы этих устройств обычно вызваны нехваткой питания. Соответственно, реакцией на появление этого устройства в логе должна быть выдача соответствующего сообщения, использование резервной мощности блока питания (если она ещё не использована), а в крайнем случае, когда питания устройству не хватает совсем — выдача рекомендации о замене блока питания и остановка системы.

  Считается, что каждому основному устройству идёт 100 ватт от блока питания в момент старта системы. Появление основного устройства в логе обозначает, что на него стало поступать на 10 ватт меньше. Первоначально у блока питания есть резерв в 20 ватт; так что первые два сбоя должны быть компенсированы; далее же подаваемая на устройства мощность начинает уменьшаться. В случае, когда на устройство подаётся не более 10 ватт, Ваша программа должна выдать рекомендацию о покупке нового блока питания и остановить систему; все последующие сообщения в логе (если таковые есть) обрабатываться уже не должны.

По заданному списку имён устройств в системе, категориям, к которым они принадлежат и логу сбоев выведите требуемые рекомендации для пользователя.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров ($0 \le T \le 100$). Далее заданы сами тестовые примеры.

Первая строка каждого тестового примера состоит из четырёх целых чисел A, B, C, D, разделённых пробелами. Первые три числа задают количество устройств в первой, второй и третьей категориях соответственно, последнее задаёт количество событий в логе.

Далее следуют A строк, содержащие имена устройств из первой категории, B строк, содержащих имена устройств из второй категории, C строк, содержащих имена устройств из третьей категории, и D строк, задающих лог сбоев; каждая запись в логе представляет собой имя устройства, с которым возникли проблемы ($0 \le A, B, C, D \le 100$, имена устройств состоят из строчных латинских букв и пробелов, начинаются и заканчиваются строчной латинской буквой и имеют длину не менее 2 и не более 22).

Output

Для каждого тестового примера выведите рекомендацию системы по каждому из сбоев в порядке их появления в логе.

Для устройств первой категории выведите "`wake` (имя устройства)". Для устройств второй категории выведите "`load` (имя устройства)", если драйвер не был загружен (в случае первого после загрузки системы появления этого устройства в логе или после выгрузки) и "`unload` (имя устройства)" в противном случае.

Для устройств третьей категории выведите "`power fail on` (имя устройства)", если на устройство подаётся более 10 ватт. В противном случае выведите "`buy the new PSU`" и завершите обработку данного тестово-

го примера вне зависимости от того, остались ли события в логе. В случае неясности с форматов вывода следуйте формату из тестовых примеров.

Examples

killthepsu.in	killthepsu.out
2 1 1 2 5 video card wireless network south bridge cpu video card wireless network wireless network south bridge cpu 0 0 1 11 motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard	wake video card load wireless network unload wireless network power fail on south bridge power fail on cpu power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard buy the new PSU

Problem L. Счастливые билеты

Input file:	lucky.in
Output file:	lucky.out
Time limit:	1 секунда
Memory limit:	512 мебибайт

Папа мальчика Кости часто ездит на трамвае. И он всегда покупает билет. Для того, чтобы мальчик любил математику, папа ему рассказывает интересные факты из мира чисел.

Сегодня он рассказал задачу о счастливых билетах, но вместо операции суммирования цифр использовал умножение. Номер билета состоит из $2N$ цифр. В записи номера разрешается использовать только определенные цифры. Билет считается счастливым, если произведение его первых N цифр равно произведению последних N цифр.

Найдите общее количество счастливых билетов.

Input

В первой строке входных данных записано целое число N ($1 \le N \le 9$). Во второй строке записаны цифры, которые можно использовать в номерах билетов. Цифры записаны без пробелов и все различны.

Output

Выведите одно целое число, равное количеству счастливых билетов.

Examples

lucky.in	lucky.out
1 7325	4
9 1	1
3 123456789	7713
2 2015	64

Problem M. Мозаика

Input file:	mosaic.in
Output file:	mosaic.out
Time limit:	1 секунда
Memory limit:	512 мебибайт

Мальчик Костя очень любит не только собирать мозаики, но и с удовольствием придумывает различные игры во время складывания их обратно в коробку.

Мозаика имеет форму прямоугольника $M \times N$, каждая клетка которого содержит одну плитку. На каждой плитке написано целое число от 1 до $M \cdot N$ (на всех плитках числа различны). В собранном виде мозаика имеет следующий вид:

1	2	3	...	$N-1$	N
$N+1$	$N+2$	$N+3$	...	$2 \cdot N-1$	$2 \cdot N$
...	...	...	...	...	...
$(M-1) \cdot N+1$	$(M-1) \cdot N+2$	$(M-1) \cdot N+3$	...	$M \cdot N-1$	$M \cdot N$

Костя хочет сыграть в игру, в которой требуется убрать как можно больше плиток мозаики по определенным правилам. За один ход можно убрать любую плитку, у которой есть 4 соседних плитки (плитки считаются соседними, если они находятся в соседних клетках в одной строке или в одном столбце).

Напишите программу, которая определит максимально возможное количество плиток, которые можно убрать по описанным выше правилам. А также найдите какой-либо корректный максимальный набор плиток.

Input

В единственной строке входных данных записаны два целых числа M и N ($1 \le M, N \le 100$) — высота и ширина мозаики.

Output

В первой строке выведите максимальное количество плиток, которые Костя может убрать по описанным выше правилам. Во второй строке выведите номера плиток в порядке, в котором их следует убирать. Если существует несколько решений, выведите любое из них.

Examples

mosaic.in	mosaic.out
3 3	1 5
2 1	0
4 4	2 6 11
4 5	3 7 9 13

Problem N. New Battle Tactics

Input file:	newbattle.in
Output file:	newbattle.out
Time limit:	1 секунда
Memory limit:	512 мебибайт

Пришло время битвы войска гномов с орками. Армия гномов, состоящая из N воинов, выстроилась в шеренгу. У каждого гнома есть сила — натуральное число от 1 до N, при этом никакие два гнома не обладают одинаковой силой. Король Даин обладает силой, равной числу B.

Даин хочет разработать тактику на предстоящий бой. Он хочет, чтобы несколько (возможно, ноль) самых левых в шеренге гномов остались на месте, и несколько (возможно, ноль) самых правых в шеренге гномов остались на месте. Таким образом король хочет организовать оборону на флангах, чтобы противник не смог зайти в тыл.

Оставшиеся гномы побегут в атаку. Разумеется, Даин хочет быть среди атакующих гномов, чтобы поднять боевой дух войска. Но он не хочет сильно выделяться среди атакующей группы, а именно, он хочет иметь среднюю силу среди членов этой группы. Иными словами, если в атаку побегут $2k + 1$ гномов, то в упорядоченном списке по силе Даин хочет быть ровно k-м среди них. В частности, атакующая группа может состоять только из нечётного числа гномов.

Ваша задача — посчитать количество вариантов выбрать атакующую группу, удовлетворяющую требованиям Даина. Две группы считаются различными, если множества гномов, находящиеся в них, не совпадают.

Input

В первой строке заданы два целых положительных числа N и B ($1 \le N \le 10^5$, $1 \le B \le N$), разделённые пробелом — количество гномов в войске гномов и сила короля Даина.

Во второй строке заданы N целых положительных чисел $P_1, P_2, \ldots, P_N$ ($1 \le P_i \le N$, P_i попарно различны) — силы воинов армии гномов, перечисленных в шеренге слева направо.

Output

Выведите в единственной строке количество вариантов для атакующей группы.

Examples

newbattle.in	newbattle.out
3 2 2 3 1	2
5 3 1 2 3 4 5	3
6 1 4 3 5 2 1 6	1