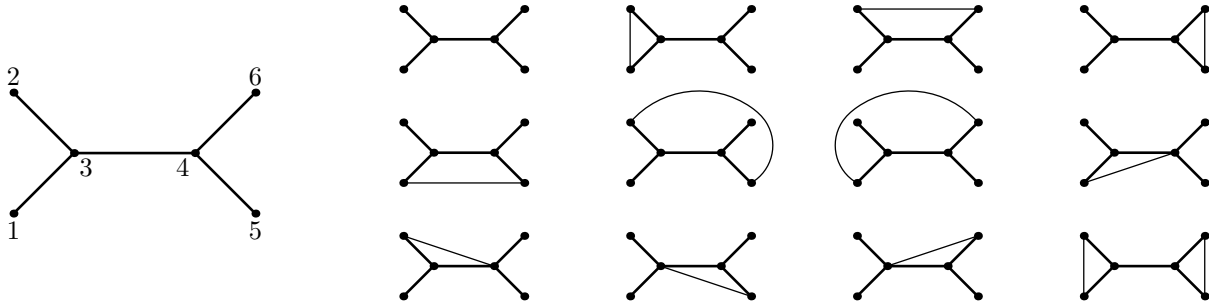
Задача A. Кактусы

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

Вершинный каутус — это связный неориентированный граф, каждая вершина которого лежит не более, чем на одном простом цикле.

Дерево можно превратить в кактус, добавив в него несколько ребер (поскольку дерево само является кактусом, то можно не добавлять ребер вообще). Вообще говоря, может существовать несколько способов превратить дерево в кактус. Количество способов сделать это назовем *кактусатостью* дерева.

Например, кактусатость дерева, изображенного на картинке слева, равна 12. Двенадцать кактусов, в которые оно может быть превращено изображены справа.



Для заданного дерева требуется найти его кактусатость.

Формат входного файла

Первая строка входного файла содержит одно целое число n — количество вершин в дереве ($1 \le n \le 200$). Следующие $n-1$ строк задают ребра дерева.

Формат выходного файла

Выведите в выходной файл единственное целое число — кактусатость заданного дерева.

Пример

cactus.in	cactus.out
6 1 3 2 3 3 4 4 5 4 6	12

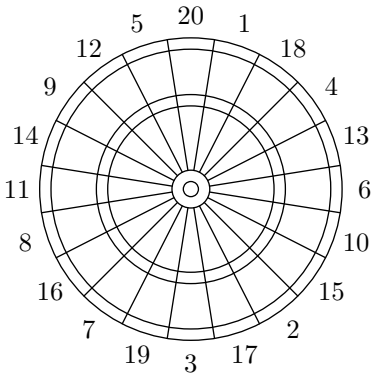
Задача B. Дартс

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

Игра в дартс очень популярна в Великобритании и Голландии. В игре принимают участие несколько игроков. Они по очереди бросают в мишень по три дротика.

В начале игры каждый игрок имеет некоторое количество очков, обычно 501. За каждое попадание дротика в мишень сумма игрока уменьшается на некоторое число, в зависимости от того, в какую часть мишени он попал. Первый, кто достигает нуля очков, считается победителем.

Внешний вид мишени показан на рисунке справа. Она разделена на 20 секторов, расположенных вокруг небольшого центрального круга. Этот круг, в свою очередь, делится на внутреннюю и внешнюю часть (иногда внутренняя часть называется *«яблочко»*). Попадание во внешнюю часть центрального круга оценивается 25 очков, а в «яблочко» — вдвое больше, то есть в 50 очков. Стоимость сектора равняется числу, которое на нем написано. Кроме того на мишени выделены два кольца — внутреннее и внешнее. Попадание в них оценивается соответственно в два и в три раза больше, чем в оставшуюся часть соответствующего сектора.



Существуют дополнительные правила для последней серии бросков, в которой игрок должен достичь нуля очков. В этой серии игроку придется бросить в мишень от одного до трех дротиков. Прави Во-первых, игрок должен достичь в точности нуля очков, получение отрицательной суммы считается ошибкой. Во-вторых, последний дротик должен быть «двойным», то есть попасть во внешнее кольцо какого-либо сектора или в «яблочко» — (оно считается удвоением внешней часть центрального круга).

Например, один из правильных способов закончить игру, имея 50 очков — бросить дротики в «`18`» и «`D16`». Способы «`D20`», «`10`», или «`20`», «`T10`» не подходят: последний бросок не является удвоенным. Еще один возможный способ победить в этом случае — просто попасть в «яблочко» («`Bull`»). По количеству оставшихся очков, найдите все способы правильно закончить игру.

Формат входного файла

Первая строка входного файла содержит число n — количество оставшихся очков ($1 \le n \le 200$).

Формат выходного файла

В первой строке выходного файла выведите k — количество способов правильно завершить партию. Каждая из следующих k строк должна содержать описание одного правильного способа. При этом число от 1 до 20 отвечает попаданию в соответствующий сектор. Буква '`D`' перед числом обозначает попадание во внешнее (удваивающее) кольцо, а '`T`' — во внутреннее (утраивающее). Внешняя часть центрального круга обозначается как «`25`», а «яблочко» (bull eye) — словом «`Bull`».

Примеры

darts.in	darts.out
5	7 1 D1 D1 1 2 D1 1 D2 D1 1 D1 T1 D1 2 1 D1 3 D1

Задача C. Домино в казино

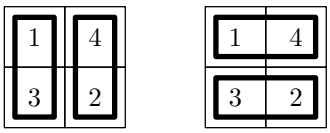
Имя входного файла:	`domino.in`
Имя выходного файла:	`domino.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Домино известно в качестве игры, в которую люди обычно играют во дворе, чтобы расслабиться после рабочего дня. Но так было лишь до того времени, пока Джон Бигбак не предоставил возможность играть в домино в своем казино «*BUMP*» (Bring Us Money, Please).

Обычная игра в домино не совсем подходит для казино, поэтому Джон создал свою собственную игру. Партия играется на прямоугольной доске размера $m \times n$. В каждой ее клетке записано некоторое целое число.

У игрока есть k костей домино — прямоугольников 2×1. Он кладет их на доску так, чтобы не возникало наложений, и его выигрыш вычисляется как сумма произведений чисел, накрытых каждой из костей домино.

Например, существует два способа положить две кости домино на доску размера 2×2. Для доски, приведенной ниже, лучший способ положить домино показан слева — в этом случае сумма составляет $1 \times 3 + 4 \times 2 = 11$. Если игрок выберет способ, показанный справа, то сумма составит $1 \times 4 + 3 \times 2 = 10$, что меньше чем 11.



По заданному расположению чисел на доске и количеству костей домино, которыми располагает игрок, найдите наибольшую сумму, которую он может получить.

Формат входного файла

Первая строка входного файла содержит целые числа m, n и k ($1 \le m \le 16$, $1 \le n \le 100$, $1 \le k \le 200$). Следующие m строк содержат по n целых чисел каждая и описывают доску. Числа, записанные на доске, неотрицательны и по величине не превосходят 1000. Гарантируется, что существует хотя бы один способ разместить все кости домино на доске.

Формат выходного файла

Выведите одно целое число — наибольшую сумму, которую может получить игрок.

Примеры

domino.in	domino.out
2 2 2 1 4 3 2	11

Задача D. Про любовь

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

Ваське нравится Машка. Она симпатичная, и он очень любит смотреть на нее. Но ему нравится и Ленка! Она тоже довольно симпатичная. Васька может смотреть на нее часами, когда она сидит на скамейке и читает книгу. Почему бы не подойти и поговорить с ней? Есть небольшая проблема. Васька — кот.

Но надо сказать, это его не очень беспокоит. В общем то, он даже привык к этому. Ему просто нравится смотреть на девочек. К сожалению, часто дома и другие препятствия не дают увидеть обеих девочек одновременно. А даже когда это можно сделать, найти подходящую точку довольно сложно. Васька просит вас, своего любимого хозяина, помочь ему.

Даны положения препятствий и точки, в которых находятся девочки. Найдите точку, из которой видны обе девочки, или установите, что такой точки нет. Конечно, Васька не может забираться внутрь препятствий.

Формат входного файла

Первая строка входного файла содержит два целых числа x_1 и y_1 — координаты Машки. Следующая строка содержит x_2 и y_2 — координаты Ленки. Третья строка входного файла содержит n — количество препятствий ($0 \le n \le 10$).

Все препятствия являются прямоугольниками со сторонами, параллельными осям координат. Каждая из следующих n строк содержит четыре целых числа $x_{i,1}$, $y_{i,1}$, $x_{i,2}$ и $y_{i,2}$ — координаты левого нижнего и правого верхнего углов препятствий. Все координаты не превосходят 100 по абсолютному значению. Препятствия не пересекаются, но могут касаться друг друга. Если два препятствия касаются друг друга углами или сторонами, между ними нет зазора. В противном случае Васька может смотреть таким образом, что линия его взгляда касается угла или идет вдоль стороны препятствия.

Ни одна из девочек не находится внутри или на границе какого-то из препятствий. Девочки находятся в разных точках.

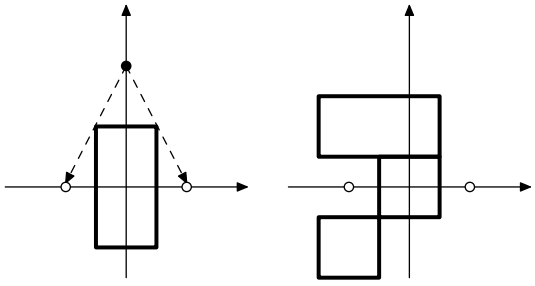
Формат выходного файла

Если точка, из которой Васька может видеть обеих девочек, существует, выведите "`YES`" на первой строке выходного файла. В этом случае вторая строка должна содержать два вещественных числа — координаты точки, из которой должен смотреть Васька. Эта точка не должна быть внутри какого-либо препятствия, но может быть на его границе. Васька не должен находиться в точке, которая одновременно принадлежит углам двух зданий, не имеющих общей стороны. Координаты должны быть выведены с точностью не менее, чем 10^{-6}.

Если искомой точки нет, выведите "`NO`" на первой строке выходного файла.

Примеры

love.in	love.out
2 0 -2 0 1 -1 -2 1 2	YES 0.0 4.0
2 0 -2 0 3 -1 -1 1 1 -3 -3 -1 -1 -3 1 1 3	NO



Задача E. Разбиения множества

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

Рассмотрим множество, состоящее из первых n натуральных чисел : $N_n = \{1, 2, \ldots, n\}$. *Разбиение* — это представление этого множества в виде объединения одного или нескольких непустых множеств. Примерами разбиений для $n = 5$ являются:

$$\{1,2,3,4,5\} = \{1,2,3\} \cup \{4,5\}$$
$$\{1,2,3,4,5\} = \{1,3,5\} \cup \{2,4\}$$
$$\{1,2,3,4,5\} = \{1,2,3,4,5\}$$
$$\{1,2,3,4,5\} = \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\}$$

Всего существует 52 разбиения множества N_5. Заметим, что разбиения, отличающиеся только порядком объединяемых множеств, не различаются.

Разбиения множества N_n можно упорядочить *лексикографически*.

Для того, чтобы определить этот порядок, вначале определим лексикографический порядок на подмножествах N_n. Будем говорить, что множество $A \subset N_n$ лексикографически меньше множества $B \subset N_n$ и писать $A < B$, если верно одно из следующих утверждений:

- найдется i, такое что $i \in A$, $i \notin B$, для всех $j < i$: $j \in A$ iff $j \in B$, и найдется $k > i$, такое что $k \in B$;
- $A \subset B$ и $i < j$ для всех $i \in A$ и $j \in B \setminus A$.

Очевидно, что введенное отношение является полным порядком на подмножествах множества N_n.

Теперь определим *каноническое представление* разбиения, как представлние, в котором объединяемые множества упорядочены лексикографически.

Разбиения упорядочиваются лексикографически следующим образом. Разбиение $N_n = A_1 \cup A_2 \cup \ldots \cup A_k$ лексикографически меньше разбиения $N_n = B_1 \cup B_2 \cup \ldots \cup B_l$, если существует такое i, что $A_1 = B_1$, $A_2 = B_2$, $\ldots$, $A_{i-1} = B_{i-1}$ и $A_i < B_i$.

По разбиению множества N_n найдите следующее в лексикографическом порядке разбиение.

Формат входного файла

Входной файл содержит несколько описаний тестов. Каждое описание является каноническим представлением разбиения. Первая строка описания содержит n и k — количество элементов в разбиваемом множестве и количество частей в разбиении ($1 \le n \le 200$). Последующие k строк содержат элементы разбиения. Элементы каждого множества упорядочены по возрастанию.

Описания тестов отделены друг от друга пустыми строками. Последняя строка входного файла содержит два нуля. Этот тест не должен обрабатываться.

Сумма n по всем описаниям не превосходит 2000.

Формат выходного файла

Для каждого теста выведите следующее в лексикографическом порядке разбиение. Если разбиение во входном файле является последним в лексигорафическом порядке, выведите первое в лексикографическом порядке. Используйте тот же формат, что и во входном файле. Отделяйте разбиения друг от друга пустыми строками.

Примеры

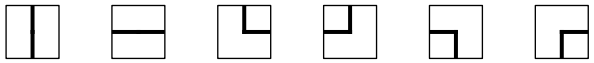
partitions.in	partitions.out
5 2 1 2 3 4 5 5 2 1 3 5 2 4 5 1 1 2 3 4 5 5 5 1 2 3 4 5 0 0	5 2 1 2 3 4 5 5 4 1 4 2 3 5 5 2 1 2 3 5 4 5 4 1 2 3 4 5

Задача F. Прокладка труб

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

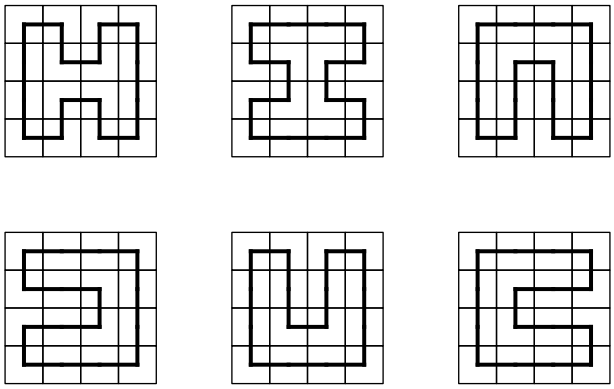
В некоторых районах города строится система центрального отопления. Каждый район города имеет форму прямоугольника и состоит из квадратных кварталов. Система центрального отопления каждого из районов представляет собой замкнутую трубу.

Чтобы сантехникам не приходилось скучать, мэр города хочет, чтобы в каждом районе трубы были уложены по-разному. Труба в каждом районе должна быть уложена таким образом, чтобы она проходила через каждый квартал. При этом существует шесть способов проложить трубу внутри квартала:



Для того, чтобы спланировать работы, мэр хочет узнать, сколькими способами можно проложить трубу в каждом районе.

Например, существует ровно 6 различных способов проложить трубу в районе из 16 кварталов, имеющего форму квадрата 4 на 4:



Формат входного файла

Входной файл содержит два целых числа r ($r > 1$) и c ($c > 1$) — размеры района города. Общее число кварталов в районе не превосходит 100 ($r \times c \leq 100$).

Формат выходного файла

Выведите число различных способов проложить трубу в данном районе.

Примеры

pipe.in	pipe.out
4 4	6
5 7	0
2 8	1
12 8	102283239429

Задача G. Словарные квадраты

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

Некоторые наборы из n слов длины n обладают интересным свойством — их можно расположить в клетках квадрата $n \times n$ так, что все слова набора можно прочитать как во вертикали, так и по горизонтали.

Примером такого набора слов является { “DATE”, “FIND”, “IDEA”, “NEXT” }. Их можно расположить так:

F	I	N	D
I	D	E	A
N	E	X	T
D	A	T	E

Заметьте, что каждое слово можно прочитать как по горизонтали, так и по вертикали. Такие квадраты называются *словарными квадратами*, наибольший известный словарный квадрат в английском языке имеет размер 10×10.

Рассмотрим еще один пример словарного квадрата:

C	R	A	B
R	A	R	E
A	R	T	S
B	E	S	T

Вам даны такие $2n$ слов, что из них можно построить два различных словарных квадрата размера $n \times n$. Ваша задача состоит в том, чтобы разбить эти слова на две группы, по n слов в каждой, и построить из слов каждой группы словарный квадрат.

Гарантируется, что все данные вам слова являются английскими словами (некоторые из них могут быть достаточно редкими словами, именами, или специальными терминами).

Формат входного файла

Первая строка входного содержит целое число n ($2 \le n \le 10$). Каждая из следующих $2n$ строк содержит слово, состоящее из заглавных букв латинского алфавита. Каждое слово содержит ровно n букв.

Формат выходного файла

Выведите два словарных квадрата, постоенных из данных слов. Разделите квадраты пустой строкой.

Примеры

square.in	square.out
4 ARTS BEST CRAB DATE FIND IDEA NEXT RARE	FIND IDEA NEXT DATE CRAB RARE ARTS BEST

Задача H. Подстрока

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

Рассмотрим слова, состоящие из букв «A», «B», «a» и «b». Скажем, что два слова эквивалентны, если одно может быть получено из другого с помощью следующих операций:

- удалить в любой позиции подстроку, равную Aa, aA, Bb или bB;
- добавить в любую позицию подстроку, равную Aa, aA, Bb или bB.

Например, слова $abAaBBabbA$ и $aAaBabaAbA$ эквивалентны:

$$abAaBBabbA \to abBBabbA \to aBabbA \to aAaBabbA \to aAaBabaAbA,$$

а слова $abAB$ и $baBA$ — нет.

Интересно отметить, что для произвольных слов α и β найдется такое слово γ, эквивалентное α, которое содержит β в качестве подстроки. Ваша задача — найти кратчайшее такое слово.

Формат входного файла

Первая строка входного файла содержит α. Вторая строка содержит β. Оба слова непусты и каждое из них имеет длину не больше 2000.

Формат выходного файла

Выведите одну строку, содержащую минимальное по длине слово, эквивалентное α, содержащее β в качестве подстроки. Если решений несколько, выведите любое.

Примеры

subword.in	subword.out
abAaBBabbA AaBaba	aAaBabaAbA

Задача I. Таблица HTML

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

При разработке программ для просмотра веб-страниц одной из наиболее сложных задач является корректное отображение таблиц. Компания «*Kozilla*», в которой вы работаете, планирует разработать новую модель браузера «*Waterrat*», и просит вас написать фрагмент ядра отображения веб-страниц, ответственный за формирование структуры таблиц.

Рассмотрим упрощенную версию фрагмента языка HTML, ответственного за описание таблиц. HTML-документ представляет собой последовательность открывающих и закрывающих *тегов*, которые ограничивают элементы документа. Внутри тегов и между ними может находится текст, который представляет собой содержимое веб-страницы.

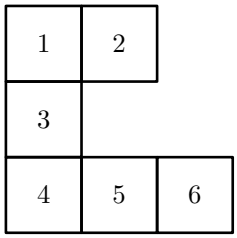
Открывающий представляет собой последовательность символов вида <*имя-тега атрибуты*>, закрывающий — </*имя тега*>. Здесь атрибуты — последовательность описаний вида *имя*="*значение*", атрибуты позволяют указать различные свойства элемента веб-страницы.

Таблица представляет собой набор ячеек, упорядоченных по строкам и столбцам. Для задания таблицы используется пара тегов <table> — </table>. Между ними находится описание таблицы.

Описание таблицы представляет собой последовательность описаний строк таблицы. Описание каждой строки представляет собой пару тегов <tr> — </tr>, между которыми расположено описание последовательности ячеек этой строки. Описание ячейки представляет собой пару тегов <td> — </td>, между которыми находится текст, который должен отображаться в этой ячейке.

Приведем пример описания таблицы и того, как она может отображаться в браузере.

```
<table>
<TR><td>1</td><td>2</td></TR>
<TR><td>3</td></TR>
<TR><td>4</td><td>5</td><td>6</td></TR>
</table>
```



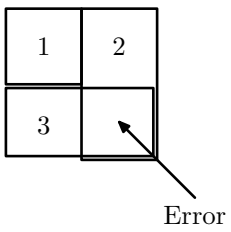
Заметим, что некоторые ячейки в некоторых строках могут отсутствовать.

Для рисования более сложных таблиц используется возможность рисовать ячейку, занимающую более одной строки или стоблца. А именно, у тега <td> могут быть атрибуты `rowspan` и `colspan`, задающие количество строк и столбцов, которые занимает ячейка, соответственно. При этом описание соответствующей ячейки размещается в описании таблицы в том месте, где располагалось бы описание ее верхнего левого угла.

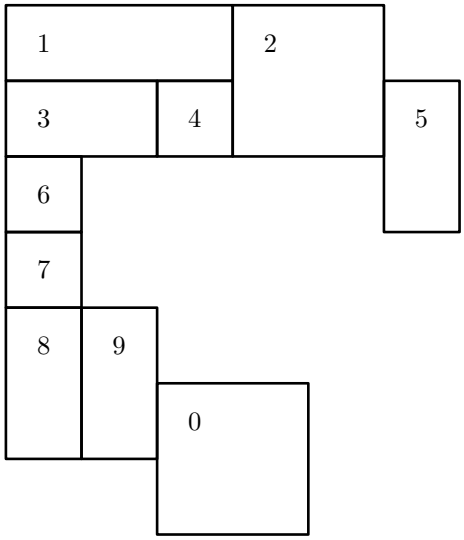
Таблица формируется строка за строкой. Если в процессе формирования строки очередная ячейка не помещается в строку, поскольку ей мешает некоторая ячейка X одной из предыдущих строк, считается что произошла ошибка.

Рассмотрим еще два примера.

```
<table>
<tr>
  <td>1</td>
  <td rowspan="2">2</td>
</tr>
<tr>
  <td colspan="2»3</td>
</tr>
</table>
```



```
<table>
<tr>
  <td colspan="3»1</td>
  <td rowspan="2" colspan="2">2</td>
</tr>
<tr>
  <td colspan="2»3</td>
  <td>4</td>
  <td rowspan="2">5</td>
</tr>
<tr>
  <td>6</td>
</tr>
<tr>
  <td>7</td>
</tr>
<tr>
  <td rowspan="2">8</td>
  <td rowspan="2">9</td>
</tr>
<tr>
  <td colspan="2" rowspan="2">0</td>
</tr>
</table>
```



По заданному описанию таблицы вам требуется вывести в выходной файл ее изображение. При этом следует игнорировать содержимое ячеек, выводя вместо него в качестве содержимого каждой ячейки пробелы.

Формат входного файла

Входной файл содержит фрагмент веб-страницы, который описывает таблицу. Гарантируется, что описание корректно и соответствует в точности одной таблице. Входной файл содержит только следующие теги: `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>` и `</td>`. Только теги `<td>` могут иметь атрибуты, при этом используются только атрибуты `rowspan` и `colspan`. При распознавании имен тегов и атрибутов следует игнорировать регистр букв. Значения атрибутов `rowspan` и `colspan` — целые положительные числа, не превосходящие 10.

Таблицы содержит по крайней мере одну строку, описание каждой строки содержит описание по крайней мере одной ячейки.

Внутри пары тегов `<td>` — `</td>` может находиться произвольный текст, состоящий из символов с ASCII-кодами от 32 до 126, кроме символа «<». Между другими тегами, а также внутри тега между названием тега и названием атрибута, между атрибутами, по обе стороны от знака «=» и между кавычкой, следующей после значения последнего атрибута, и символом «>» может быть произвольное число пробельных символов — пробелов и символов перевода строки. Их следует игнорировать.

Гарантируется, что описанная таблица содержит не более 100 ячеек, размер входного файла не превышает 10 килобайт.

Формат выходного файла

Выведите в выходной файл изображение таблицы.

Используйте следующие символы:

‘+’ (плюс) — для отображения пересечения линий;
‘|’ (вертикальная черта, код ASCII 124) — для отображения вертикальных линий;
‘–’ (минус) — для отображения горизонтальных линий;
‘ ’ (пробел) — во всех остальных случаях;

Ячейка, занимающая m строк и n столбцов должна быть изображена с использованием $2m - 1$ строки по $2n - 1$ пробела.

Если в описании таблицы содержатся ошибка, выведите “ERROR” в выходной файл.

Примеры

table.in	table.out
<table> <TR><td>1</td><td>2</td></TR> <TR><td>3</td></TR> <TR><td>4</td><td>5</td><td>6</td></TR> </table>	+-+-+ | | | +-+-+ | | +-+-+-+ | | | | +-+-+-+
<table> <tr> <td>1</td> <td rowspan="2">2</td> </tr> <tr> <td colspan="2»3</td> </tr> </table>	ERROR
<table> <tr> <td colspan="3»1</td> <td rowspan="2" colspan="2">2</td> </tr> <tr> <td colspan="2»3</td> <td>4</td> <td rowspan="2">5</td> </tr> <tr> <td>6</td> </tr> <tr> <td>7</td> </tr> <tr> <td rowspan="2">8</td> <td rowspan="2">9</td> </tr> <tr> <td colspan="2" rowspan="2">0</td> </tr> </table>	+-----+---+ | | | +---+-+ +-+ | | | | | +-+-+-+---+ | | | | | +-+ +-+ | | +-+-+ | | | | | +---+ | | | | +-+-+ | | | +---+

Задача J. Поле чудес

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

Всем известно сверхпопулярное теле-шоу «*Поле чудес*».

В игре принимают участие n человек. Их цель — отгадать загаданное слово. Изначально известна только длина слова l, а буквы на специальном табло закрыты черными прямоугольниками.

В свой ход игрок называет одну букву. Если эта буква встречается в слове, то все ее вхождения открываются, и игрок делает еще один ход. Если в слове нет такой буквы, то ход передается следующему игроку. От последнего игрока ход передается первому. Выигрывает тот игрок, который отгадывает последнюю букву.

Пусть, например, загадано секретное слово «`CONTEST`». Изначально игроки видят только «`-------`». Если первый игрок скажет '`E`', то она открывается, и теперь табло выглядит как «`----E--`». Первый игрок делает еще один ход — пусть, например, он называет '`A`'. В слове нет такой буквы, поэтому ход передается следующему игроку. Если тот скажет '`T`', то игроки увидят «`---TE-T`», и так далее.

Пашин друг собирается поучаствовать в игре. Паше интересно, каковы шансы на победу его друга. Паша оценил интеллектуальный потенциал каждого из игроков q_t. Вероятность того, что игрок t правильно отгадает букву в ситуации, когда еще не были названы i букв, неизвестны еще j разных букв слова, и на табло осталось k закрытых букв, вычисляется по следующей формуле:

$$p_{t,i,j,k} = \left(1 - q_t^{\frac{1}{k-j+1}}\right)\frac{j}{i} + q_t^{\frac{1}{k-j+1}}\left(1 - \frac{1}{i}\right)^{i-j}.$$

Здесь мы будем считать, что $0^0 = 1$.

Вероятности угадывания каждой из все еще неизвестных букв слова одинаковы.

По заданным n, очереди хода Пашиного друга r, загаданному слову и значениям интеллектуального потенциала игроков, определите вероятность того, что Пашин друг победит в игре.

Формат входного файла

Первая строка входного файла содержит n и r ($2 \le n \le 10$, $1 \le r \le n$). На второй строке записано загаданное слово. Оно состоит из больших букв латинского алфавита, и его длина не превосходит 12. Третья строка входного файла содержит n вещественных чисел — значения интеллектуального потенциала игроков ($0 \le q_t \le 0.99$).

Формат выходного файла

Выведите одно число — вероятность победы Пашиного друга. Ответ должен быть дан с точностью 10^{-8}.

Примеры

wheel.in	wheel.out
3 1 CONTEST 0.7 0.2 0.1	0.4648222937