

Задача А. Рудольф и дрессировка

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

У Рудольфа есть любимая собачка, которую зовут Байер. Как и всякий ответственный хозяин, Рудольф старается воспитывать своего питомца и учит его выполнять различные команды. Однако традиционные трюки вроде «*Сидеть*» и «*Апорт*» не очень-то и впечатляют Рудольфа, поэтому он придумал для Байера кое-что получше. Рудольф — большой фанат структур данных, в связи с чем он предлагает Байеру рисовать красно-чёрные деревья.

Красно-чёрное дерево — это древовидная структура данных, имеющая следующие особенности:

- узлы дерева делятся на внутренние и листовые;
- каждый внутренний узел имеет уникальный целочисленный ключ и двух потомков; листовые узлы ключей и потомков не имеют;
- по ключам внутренние узлы образуют двоичное дерево поиска;
- каждый узел имеет цвет — красный или чёрный;
- корень дерева и листовые узлы являются чёрными;
- родитель любого красного узла является чёрным;
- все простые пути от любого узла до листовых узлов, являющихся его потомками, содержат одинаковое количество чёрных узлов.

Сегодня Рудольф решил пообедать в своём любимом кафе. К сожалению, туда не пускают с животными, поэтому Байеру придётся остаться дома. Чтобы питомцу не было грустно, Рудольф поручил ему составить примеры красно-чёрных деревьев. Байер с радостью согласился и с энтузиазмом взялся за работу.

Через три часа Рудольф вернулся из кафе сытый и довольный. К своему удивлению, он обнаружил, что Байер успел нарисовать огромное количество красно-чёрных деревьев: среди них были маленькие, средние, большие и даже очень большие. Рудольф смог визуально оценить только деревья небольшого размера и нашёл несколько ошибок. А вот проверить остальные деревья оказалось очень сложно, поэтому Рудольф попросил вас автоматизировать оценку трудов Байера.

Формат входных данных

Первая строка содержит число n ($1 \leq n \leq 10^5$) — количество внутренних узлов красно-чёрного дерева.

Следующие n строк описывают внутренние узлы. Каждая из них содержит строку c_i , целое число k_i и целое число p_i ($c_i \in \{red, black\}$, $1 \leq k_i, p_i \leq 10^{18}$) — цвет узла, ключ узла и ключ родителя, соответственно. Для корневого узла $p_i = -1$.

Гарантируется, что описанный во входных данных граф является деревом и имеет корень.

Формат выходных данных

Выведите «YES» (без кавычек), если входные данные содержат корректное описание красно-чёрного дерева (с учётом добавления листовых узлов). В противном случае выведите «NO» (без кавычек).

Примеры

input.txt	output.txt
5 red 30 40 black 20 -1 black 10 20 black 40 20 red 50 40	YES
5 red 3 4 red 5 4 black 2 -1 black 1 2 red 4 2	NO

Задача В. Рудольф и книжные шкафы

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Наконец-то Рудольф открыл копилку и приобрёл Сверхбольшую Энциклопедию Структур Данных в n томах. Теперь перед ним встала проблема её размещения: энциклопедия действительно весьма велика и может не поместиться даже в целый шкаф, не говоря уже о том, чтобы все её тома можно было поставить на одну полку.

Рудольф попросил своих знакомых Леонидаса и Роберта смастерить шкафы для энциклопедии, но им нужно точно знать, сколько шкафов необходимо изготовить. Исходя из размеров комнаты, Роберт смог подсчитать, что в один шкаф можно уместить l полок, а на одной полке можно расположить k томов. При этом необходимо, чтобы количество шкафов было минимально, то есть каждый шкаф должен вмещать максимально возможное количество томов. Тем не менее, последний шкаф может быть заполнен не полностью.

Теперь осталось самое сложное — определить количество шкафов, необходимых для хранения всех томов энциклопедии. Помогите Рудольфу решить эту задачу.

Формат входных данных

Ввод содержит целые числа n , l и k ($1 \leq n \leq 10^{12}$, $1 \leq l \leq 1000$, $1 \leq k \leq 10^6$) — соответственно количество томов энциклопедии, количество полок в одном шкафу и количество томов, которые умещаются на одной полке.

Формат выходных данных

Выведите одно целое число — количество шкафов, необходимых для хранения энциклопедии.

Примеры

<code>input.txt</code>	<code>output.txt</code>
10 2 3	2
10 2 5	1

Задача С. Рудольф и правильный порядок

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Теперь, когда количество шкафов посчитано, и Леонидас и Роберт смогли их изготовить, перед Рудольфом встала не менее важная и сложная проблема: такая важная энциклопедия, как Сверхбольшая Энциклопедия Структур Данных, не может стоять абы как, её тома должны стоять в *правильном* порядке, чтобы их было удобно искать и читать.

Томы нумеруются различными целыми числами от 1 до n . *Правильный* порядок предполагает, что тома расставлены по возрастанию количества единичных бит в двоичном представлении их номеров, а если для нескольких томов это количество совпадает — по возрастанию номеров томов.

Томы энциклопедии ставятся в шкаф начиная с верхней полки, слева направо. Том ставится в следующий шкаф только в том случае, если предыдущий шкаф полностью заполнен.

Вручную определять порядок томов оказалось утомительным даже для такого сильного математика, как Рудольф. Помогите ему написать программу, которая подскажет, как расставить тома энциклопедии в конкретном шкафу.

Формат входных данных

Первая строка содержит целые числа n , l и k ($1 \leq n \leq 1000$, $1 \leq l \leq 100$, $1 \leq k \leq 100$) — соответственно количество томов энциклопедии, количество полок в одном шкафу и количество томов, которые умещаются на одной полке.

Вторая строка содержит целое число m ($1 \leq m \leq 1000$) — номер шкафа, который интересуется Рудольфа. Гарантируется, что шкаф m будет заполнен полностью или частично.

Формат выходных данных

Выведите одну или более строк, каждая из которых содержит номера томов энциклопедии, которые будут стоять в шкафу m , в «*правильном*» порядке.

Примеры

input.txt	output.txt
10 2 3 2	6 9 10 7
6 2 2 1	1 2 4 3

Задача D. Рудольф и очереди

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Давайте немного отвлечёмся от красно-чёрных деревьев, перенесёмся в любимое кафе Рудольфа и выясним, почему же Рудольф так долго трапезничал. Кафе работает по принципу шведского стола: можно выбирать любые блюда, а затем подходить к кассе и оплачивать их. Имеются две кассы, к каждой из которых ведёт отдельная очередь.

Рудольф зашёл в кафе и понял, что попал как раз в обеденное время — у обеих касс были огромные очереди.

Будучи грамотным математиком, Рудольф сразу же посчитал, что в первой очереди n_1 человек, а во второй — n_2 . Следуя логике, он встал в очередь с наименьшим количеством людей. Спустя некоторое время он понял, что очереди двигаются с разными скоростями, поэтому выбор наименьшей очереди вовсе не означал, что Рудольф быстрее оплатит свой обед. Рудольф заметил, что скорость обслуживания зависит от возраста обслуживаемого человека — старшему поколению гораздо сложнее разобраться с хитроумными устройствами безналичной оплаты, чем молодым людям. Рудольф разделил всех людей на категории в соответствии с их возрастом. Для каждой категории Рудольф определил время обслуживания на кассе:

- 18–35 лет — 1 минута,
- 36–45 лет — 3 минуты,
- 46–65 лет — 5 минут,
- 66–100 лет — 10 минут.

Поскольку Рудольф хочет как можно быстрее оплатить свой обед, он решил, что каждую минуту с вероятностью p он будет переходить в другую очередь, если она на момент принятия решения состоит из меньшего числа людей.

Рудольфу стало интересно, чему равно математическое ожидание количества переходов между очередями за период времени с момента занятия очереди в кафе до момента оплаты обеда. Давайте поможем ему в решении этой задачи.

Считать, что остальные люди не перемещаются между очередями, а также что в очередях не появляются новые люди.

Формат входных данных

Первая строка содержит вещественное число p ($0 \leq p \leq 1$), заданное не более чем с 6 знаками после десятичной точки, — вероятность перехода Рудольфа из одной очереди в другую.

Вторая строка содержит целое число n_1 ($1 \leq n_1 \leq 100$) — количество людей в первой очереди.

Третья строка содержит n_1 целых чисел a_{1i} ($18 \leq a_{1i} \leq 100$) — возраст людей в первой очереди в порядке обслуживания.

Четвертая строка содержит целое число n_2 ($1 \leq n_2 \leq 100, n_2 \neq n_1$) — количество людей во второй очереди.

Пятая строка содержит n_2 целых чисел a_{2j} ($18 \leq a_{2j} \leq 100$) — возраст людей во второй очереди в порядке обслуживания.

Формат выходных данных

Выведите одно вещественное число — математическое ожидание количества переходов между очередями, которые может совершить Рудольф при заданных условиях. Абсолютная или относительная погрешность ответа не должна превышать 10^{-6} .

Примеры

input.txt	output.txt
1.0 1 70 5 18 18 18 18 18	1.000000000
1.0 3 18 70 18 5 18 18 18 18 18	1.000000000
0.5 3 18 18 18 2 18 18	0.000000000
0.5 3 20 50 30 4 18 30 40 18	0.750000000

Задача Е. Рудольф и квазирегулярный граф

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Для своих научных изысканий Рудольф часто штудировал книги по теории графов и находит в них интересные определения: например, недавно он узнал, что неориентированный граф, у которого все вершины имеют равное количество рёбер-соседей, называют регулярным.

Иногда Рудольф вводит собственные термины, если имеющиеся ему чем-то не угодили. Например, ему не понравилось, что понятие регулярности характерно только для неориентированных графов. Кроме того, ограничение, согласно которому из каждой вершины исходит строго одинаковое количество рёбер, показалось ему слишком скучным. Поэтому Рудольф придумал определение квазирегулярного графа.

Ориентированный граф является квазирегулярным, если он обладает следующими свойствами:

1. количество рёбер, исходящих из каждой вершины, либо равно нулю, либо кратно некоторому целому числу $k > 1$ (разумеется, k постоянно);
2. граф не содержит петель (рёбер, ведущих из вершины в неё же) и кратных рёбер (нескольких рёбер, исходящих из одинаковых вершин и входящих в одинаковые вершины).

Рудольф остался очень доволен своим новым определением и тут же начал рисовать различные квазирегулярные графы. Спустя некоторое время он обратил внимание на то, что некоторые из нарисованных им графов содержали циклы, тогда как другие — нет. Из книг Рудольф знал, что ациклические графы — это очень здорово, поэтому он принялся подробнее изучать (и рисовать) именно ациклические квазирегулярные графы.

Сейчас Рудольф нарисовал на бумаге n точек-вершин. Ему интересно, какое максимальное количество стрелок-рёбер он может дорисовать, чтобы получившийся граф был квазирегулярным и не содержал циклов. Помогите ему найти ответ на этот вопрос.

Формат входных данных

Ввод содержит целое число n ($1 \leq n \leq 10^9$) — количество вершин графа.

Формат выходных данных

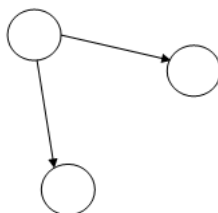
Выведите одно целое число — максимальное количество рёбер ациклического квазирегулярного графа, содержащего n вершин.

Примеры

<code>input.txt</code>	<code>output.txt</code>
3	2
6	12

Замечание

Искомый ациклический квазирегулярный граф для первого примера показан на рисунке.



Задача F. Рудольф и декодирование

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Для защиты своих наиболее важных научных работ от посторонних посягательств Рудольф решил разработать собственный шифр.

Тексты работ Рудольфа включают n различных символов. Для каждого из этих символов Рудольф придумал определённый код — последовательность бит, на которую указанный символ должен быть заменён при шифровании. Например, если символам C, D, E и O соответствуют коды 0, 1, 10 и 11, то слово «CODE» в зашифрованном виде выглядит так: 011110.

Экспериментируя с получившимся шифром, Рудольф, к своему ужасу, обнаружил, что иногда сообщение становится невозможно восстановить! Например, показанная выше последовательность 011110 может быть декодирована не только как «CODE», но и как «CDOE», и даже как «CDDDDC».

Охваченный паникой, Рудольф придумал новые коды символов, но он всё ещё опасается, что и этот шифр может оказаться неподходящим. Помогите Рудольфу определить, всегда ли его шифр можно декодировать однозначно.

Формат входных данных

Первая строка содержит целое число n ($1 \leq n \leq 100$) — количество различных символов в текстах Рудольфа.

Следующие n строк описывают коды символов. Каждая из них содержит строку s_i ($1 \leq |s_i| \leq 20$), состоящую из цифр 0 и 1, на которую i -й символ алфавита заменяется при шифровании.

Формат выходных данных

Если любое зашифрованное сообщение, составленное из данных кодов, может быть декодировано единственным образом, выведите «YES» (без кавычек). В противном случае, выведите «NO» (без кавычек).

Примеры

input.txt	output.txt
4 0 1 10 11	NO
4 00 001 10 11	YES

Задача G. Рудольф и раскраска

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Рудольф немного устал от математики и решил отвлечься. В закромах своего чердака он нашёл старую раскраску, которая оказалась весьма необычной.

Каждая страница раскраски представляет собой прямоугольную сетку размера $m \times n$. Левый верхний угол сетки имеет координаты $(0; 0)$, ось Ox направлена слева направо, ось Oy — сверху вниз. А на обратной стороне страницы напечатана инструкция в виде последовательности прямоугольных областей и цветов, в которые надо их покрасить.

Путём нехитрых подсчётов Рудольф выяснил, что в инструкциях по раскрашиванию фигурируют ровно p цветов. Недолго думая, он разыскал нужные краски и приступил к работе.

В процессе раскрашивания Рудольф заметил, что некоторые области, приведённые в инструкции, пересекаются. И если перемешать краски двух цветов, то в результате получится некоторый новый цвет. На последней странице раскраски Рудольф обнаружил палитру, которая для любой пары цветов позволяет определить, какой цвет получится в результате их перемешивания.

Рудольф ничего не смог поделать со своей тягой к автоматизации. Ему стало интересно, как реализовать программу, раскрашивающую прямоугольную сетку по имеющейся инструкции и карте соответствия цветов.

Гарантируется, что результат работы Рудольфа **не зависит** от порядка, в котором он будет закрашивать области.

Формат входных данных

Первая строка содержит целые числа n и m ($1 \leq n, m \leq 2000$) — количество столбцов и строк прямоугольной сетки раскраски.

Вторая строка содержит целое число p ($1 \leq p \leq 26$) — количество цветов.

Следующие строки описывают результаты перемешивания всех возможных пар цветов. Каждая из этих строк содержит символы c_{ai} , c_{bi} и c_{ri} ($c_{ai}, c_{bi}, c_{ri} \in \{a, \dots, z\}$) — идентификаторы двух перемешиваемых цветов и результирующего цвета соответственно. Гарантируется, что в описании по одному разу присутствует каждая пара цветов.

Следующая строка содержит целое число k ($0 \leq k \leq 1000$) — количество прямоугольных областей в инструкции по раскрашиванию.

Следующие k строк описывают инструкцию по раскрашиванию. Каждая из них содержит целые числа x_1, y_1, x_2, y_2 , а также символ c_j ($0 \leq x_1 \leq x_2 \leq n, 0 \leq y_1 \leq y_2 \leq m, c_j \in \{a, \dots, z\}$) — координаты левого верхнего и правого нижнего углов прямоугольной области и идентификатор цвета, которым нужно закрасить данную прямоугольную область, соответственно. Гарантируется, что в инструкции используются только те цвета, которые есть в палитре.

Формат выходных данных

Выведите m строк по n символов. Каждый символ должен соответствовать результирующему цвету клетки прямоугольной сетки, соответствующей позиции символа.

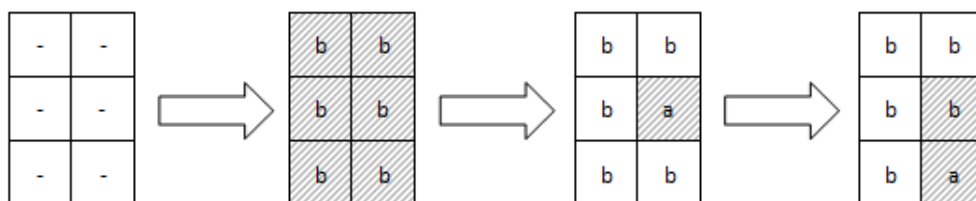
Если в результате на сетке останутся неокрашенные области, в соответствующих позициях выведите символ «-». Изначально сетка не окрашена.

Примеры

input.txt	output.txt
2 3 2 a b a a a b b b b 3 0 0 2 3 b 1 1 2 2 a 1 1 2 3 a	bb bb ba
3 3 4 y b g c c c b g c c g g b c b y y c g y b y c y b b y g g y 3 1 2 3 3 g 1 0 3 2 c 1 1 2 2 g	-cc -gc -gg

Замечание

В первом примере необходимо раскрасить сетку, состоящую из двух столбцов и трёх строк. В распоряжении имеются два цвета и палитра, состоящая из трёх правил смешивания цветов. Последовательность раскраски сетки показана на рисунке.



Задача Н. Рудольф и забор

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Как-то раз, проснувшись с утра и обнаружив, что у него украли дом, Рудольф понял, что ему необходимо защитить свой участок от подобных вторжений. Он решил построить забор и для этого закупил в строительном магазине доски. Когда он вернулся из магазина и осмотрел свою покупку, он понял, что не все доски имеют одинаковую длину. Тогда Рудольф решил выйти из ситуации самым очевидным образом, а именно — построить симметричный забор.

Забор из n досок Рудольф считает симметричным, если доски в позициях i и $(n - i + 1)$, где $(1 \leq i \leq \frac{n}{2})$, имеют равные высоты.

Ко всему прочему Рудольф хочет, чтобы его забор имел максимальную длину. Однако, он внезапно осознал, что способов построить такой забор может быть очень много, и попросил вас помочь ему подсчитать их количество. Два способа построения забора считаются различными, если высоты досок отличаются хотя бы в одной позиции.

Помогите Рудольфу узнать ответ на его вопрос.

Формат входных данных

Первая строка содержит целое число n ($1 \leq n \leq 10^5$) — количество досок у Рудольфа.

Вторая строка содержит n целых чисел h_i ($1 \leq h_i \leq 10^{18}$) — высоту каждой из досок.

Формат выходных данных

Выведите одно целое число — количество различных способов составить симметричный забор максимальной длины из имеющихся у Рудольфа досок. Так как это количество может оказаться достаточно большим, выведите остаток от его деления на $(10^9 + 9)$.

Примеры

<code>input.txt</code>	<code>output.txt</code>
5 2 1 2 2 1	2
6 1 3 2 2 1 1	4

Замечание

В первом примере существуют два способа составления симметричного забора из всех имеющихся досок: $(1, 2, 2, 2, 1)$ и $(2, 1, 2, 1, 2)$.

Во втором примере для получения максимально длинного забора мы можем задействовать только 5 досок и составить следующие симметричные заборы: $(1, 2, 1, 2, 1)$, $(1, 2, 3, 2, 1)$, $(2, 1, 1, 1, 2)$, $(2, 1, 3, 1, 2)$.

Задача I. Рудольф и ДНК

Имя входного файла: `input.txt`
Имя выходного файла: `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Во время экспедиции в кратер Ун'горо Рудольф и его верный спутник Байер собрали множество интересных образцов ископаемой флоры и фауны. Теперь они занимаются исследованием кода ДНК элементарного динозавра в попытках определить гены, послужившие причиной элементарной мутации.

Код ДНК динозавра представляет собой строку, состоящую из n заглавных букв латинского алфавита. Предполагается, что ген элементарной мутации представляет собой максимальную по длине подстроку кода ДНК, повторяющуюся более одного раза.

Помогите Рудольфу написать программу, определяющую ген элементарной мутации.

Формат входных данных

Ввод содержит строку s ($1 \leq |s| \leq 10^5$), состоящую из заглавных букв латинского алфавита — код ДНК.

Формат выходных данных

Если в коде ДНК ген элементарной мутации отсутствует, выведите «NONE» (без кавычек).

В противном случае в первой строке выведите одно целое число — длину гена элементарной мутации.

Во второй строке выведите позиции, начиная с которых в коде ДНК расположен искомый ген. Выведенные позиции должны быть упорядочены по возрастанию, строка индексируется с нуля.

Если максимальную подстроку невозможно определить однозначно, во второй строке выведите позиции всех возможных кандидатов.

Примеры

input.txt	output.txt
EXPEDITIONUNGOROEEXPEDITION	10 0 16
ABBA	1 0 1 2 3
AB	NONE

Задача J. Рудольф и Marine Front Unlimited ZX

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Игра «Marine Front Unlimited ZX» становится очень популярной в Сети — все друзья Рудольфа только и делают, что говорят о ней! Рудольф тоже решил попробовать модную игру, но, поиграв в неё несколько минут, начал что-то подозревать. Поиграв ещё пару часов, он полностью убедился в том, что невероятная новинка — не более чем очередной клон игры «Морской бой».

Если вы вдруг не знаете, как играть в «Морской бой», то мы вам расскажем. Эта игра предназначена для двух игроков. Прежде всего каждый из них берёт клетчатый лист и очерчивает на нём два одинаковых поля, содержащих n столбцов и m строк, — одно для своих кораблей, а другое для кораблей противника. Столбцы каждого из полей помечаются слева направо первыми n буквами латинского алфавита, а строки — сверху вниз первыми m натуральными числами. Таким образом, каждая клетка имеет уникальные координаты: например, левый верхний угол обозначается как **a1**. Затем каждый из игроков втайне от другого рисует на своём поле набор кораблей — фигур, составленных из клеток, имеющих общую сторону. Форма, количество и правила размещения кораблей зависят от конкретных вариантов игры.

На этом подготовка завершается, и игроки начинают по очереди обстреливать поле противника. Один игрок называет другому координаты клетки, по которой он стреляет. Если в этой клетке нет кораблей, второй игрок отвечает «Мимо» и получает право хода. В противном случае второй игрок говорит «Попал» (или «Потопил», если противник уже расстрелял все остальные клетки корабля), а первый может стрелять ещё раз. Цель игры — первым потопить все корабли соперника.

В «Marine Front Unlimited ZX» в роли противника выступает компьютер. Рудольф заметил, что интеллект у него так себе — он всегда обстреливает клетки игрока в одном и том же порядке! Тем не менее, у игры есть и положительные стороны — например, достижения. Для получения одного из них требуется поставить на своём поле единственный четырёхпалубник (прямоугольник 1×4 или 4×1) таким образом, чтобы компьютер потопил его как можно позже. Однако и компьютер при этом меняет свою стратегию: он заведомо не стреляет по клеткам, в которых четырёхпалубник не может оказаться, а впервые попав по нему, всегда добивает его тремя следующими выстрелами.

Хоть Рудольф и не в восторге от новой игры, но получить достижение он всё-таки хочет. Помогите ему правильно поставить корабль.

Формат входных данных

Первая строка содержит целые числа n и m ($4 \leq n, m \leq 26$) — количество столбцов и строк игрового поля соответственно.

Следующие $m \times n$ строк описывают порядок, в котором компьютерный противник обстреливает клетки. Каждая из этих строк содержит координаты одной из клеток поля. Гарантируется, что все координаты уникальны.

Формат выходных данных

Выведите одно целое число — максимальное количество выстрелов, после которых размещённый некоторым образом на поле четырёхпалубник будет потоплен.

Примеры

input.txt	output.txt
5 4 a1 b1 c1 d1 e1 a2 b2 c2 d2 e2 a3 b3 c3 d3 e3 a4 b4 c4 d4 e4	14
4 4 c4 d3 d4 c3 a1 a2 a3 a4 b4 b3 b2 b1 c1 d1 c2 d2	8

Замечание

В первом примере целесообразно разместить четырёхпалубник в клетках b4, c4, d4, e4. Компьютер не будет стрелять по клеткам c2, d2, e2, c3, d3, e3, так как после обстрела предыдущих клеток ясно, что корабль в них оказаться не может.