Задача A. Распределение воздушных шаров

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В ходе подготовки «Кубка Томска по программированию 2018» возникла небольшая трудность.

За каждую решенную задачу планируется приклеивать воздушный шарик к столу команды, которая ее решила. Каждой задаче должен соответствовать шарик одного определенного цвета. Никаким двум задачам не должен соответствовать шарик одного и того же цвета.

Оргкомитет подготовил восемь задач, каждой из которых соответствует уникальная буква латинского алфавита от «A» до «H». В магазине, где планируется закупать шарики, имеются шарики ровно восьми различных цветов. Известна стоимость одного шарика каждого из цветов.

В Томске имеется N команд олимпиадных программистов. Для каждой из которых организаторам известен набор задач, которые они могут решить. Организаторы такого уровня никогда не ошибаются. Поэтому если какой-то задачи нет в наборе какой-либо команды, то эта команда точно не решит ее на соревновании.

Организаторы понимают, что бюджет олимпиады составляет всего лишь S рублей, и его может не хватить для того, чтобы купить столько шариков, сколько нужно. Поэтому, возможно, придется пригласить лишь некоторые из N команд. Причем организаторы хотят купить такой набор шариков, чтобы даже если все приглашенные команды решат все задачи, которые они могут решить, закупленных шариков все равно хватит.

Перед организаторами стоит непростая задача — распределить цвета шариков по задачам и пригласить некоторые команды так, чтобы уложиться в имеющийся бюджет и при этом пригласить как можно больше команд.

Ваша задача — написать программу, которая поможет определить максимальное возможное количество приглашенных команд.

Формат входного файла

Первая строка содержит два целых числа N и S — количество команд в Томске и размер бюджета олимпиады ($1 \leqslant N \leqslant 10^5, 1 \leqslant S \leqslant 10^9$). Во второй строке содержится восемь целых чисел от 0 до 10^4 — стоимости шариков различных цветов в магазине. Далее следует N строк, каждая из которых задает набор задач, которые может решить очередная команда. Каждая такая строка состоит из заглавных букв латинского алфавита. Буквы могут быть в произвольном порядке. Никакая буква не встречается более одного раза для одной команды.

Формат выходного файла

Выведите единственное целое число — максимальное возможное количество приглашенных команд.

Примеры

stdin	stdout
5 13 1 2 3 4 5 6 7 8 AGE AGH AG AE ABCD	3

Note

В приведенном примере следует пригласить первую, третью и четвертую команды, за задачу «A» давать шарик стоимости 1, за задачу «G» — стоимости 2, а за задачу «E» — стоимости 3.

Задача B. Число Хаустова

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В одном из университетов славного города Томска проходят сборы по программированию, в которых участвует N команд. Для удобства пронумеруем команды последовательными целыми числами от 1 до N.

Сборы длятся M дней, в каждый из которых проводится ровно одно соревнование. Каждый день все N команд принимают участие в соревновании, и каждая из них занимает какое-либо место от 1 до N. Место каждой команды уникально, то есть никакие две команды не могут разделить одно и то же место.

Один из организаторов сборов — г-н Хаустов. Он утверждает, что наблюдать за ходом этих сборов не так уж интересно, как хотелось бы. Для того, чтобы не быть голословным, г-н Хаустов посчитал важное числовое значение — число Хаустова. Оно равняется количеству очевидных пар команд среди команд-участниц сборов. Пара команд (A, B) является очевидной, если команда A заняла место выше, чем команда B в каждом из соревновательных дней.

Другие организаторы сборов хотят убедиться в том, что число Хаустова посчитано верно. Для этого им нужна программа, которая может посчитать это значение.

Ваша задача — написать программу, которая по описанию каждого из соревновательных дней определит правильное значение числа Хаустова.

Формат входного файла

Первая строка входных данных содержит два целых числа через пробел: N и M — количество команд-участниц сборов и количество соревновательных дней ($1 \leqslant N, M \leqslant 100$).

Далее следует M строк, каждая из которых описывает очередной соревновательный день. Каждое такое описание состоит из N различных целых чисел от 1 до N, числа разделяются пробелом. Первое число — номер команды, которая заняла первое место, второе число — номер команды, которая заняла второе место и так далее.

Формат выходного файла

Выведите единственное целое число — число Хаустова для заданных во входных данных сборов.

Примеры

stdin	stdout
4 3 1 2 3 4 1 3 2 4 1 3 4 2	4

Note

В приведенном примере очевидными парами являются пары: $(1, 2)$, $(1, 3)$, $(1, 4)$ и $(3, 4)$.

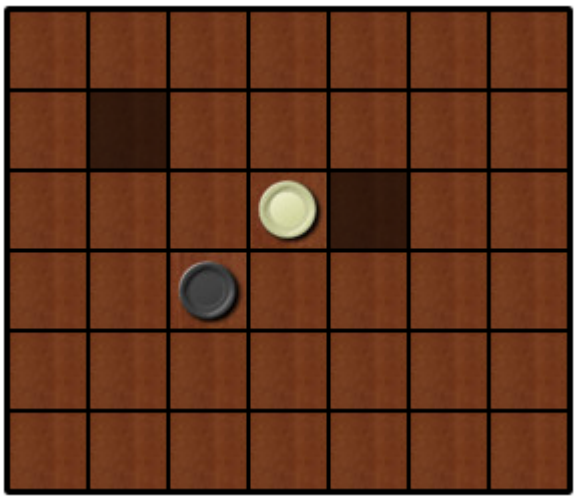
Задача C. Colonization 2013

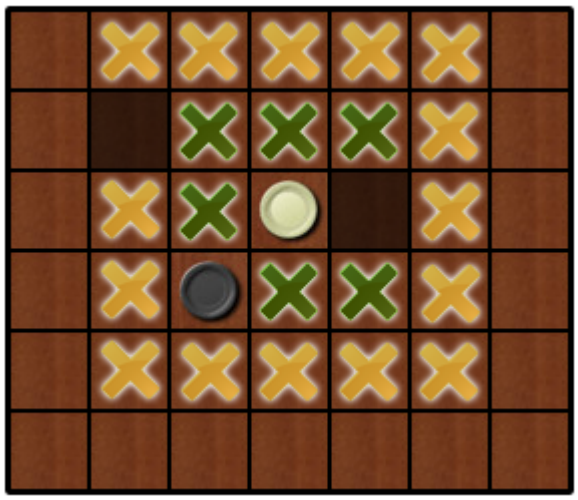
Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В 2013 году некоторую популярность набрала игра «Colonization 2013». Играется эта игра на клетчатом поле размера 12×12. Клетки нумеруются по строкам (сверху вниз) и по столбцам (слева направо) целыми числами от 1 до 12. Некоторые из клеток поля не участвуют в игре. Клетки, которые участвуют в игре будем называть активными, остальные — неактивными. На некоторых из активных клеток изначально располагаются шашки. Шашки белого цвета принадлежат первому игроку, шашки черного цвета — второму игроку. Игроки ходят по очереди. В свой ход каждый из игроков может выбрать ровно одну из своих шашек и выполнить одно из следующих действий (для определенности предположим, что выбрана шашка, стоящая на клетке (X, Y)):

- Создать на одной из клеток (X', Y') соседних для клетки (X, Y) шашку своего цвета. Клетка (X', Y') должна быть активной, и на ней не должно находиться других шашек. Такое действие будем называть «копированием».
- Перенести на одну из клеток (X', Y') почти соседних для клетки (X, Y) шашку из клетки (X, Y). Стоит отметить, что после совершения этого действия клетка (X, Y) окажется пустой. Клетка (X', Y') должна быть активной, и на ней не должно находиться других шашек. Такое действие будем называть «переносом».

Клетка (X_1, Y_1) являетя соседней для клетки (X_2, Y_2), если $max(|X_1 - X_2|, |Y_1 - Y_2|) = 1$.
Клетка (X_1, Y_1) являетя почти соседней для клетки (X_2, Y_2), если $max(|X_1 - X_2|, |Y_1 - Y_2|) = 2$.





На изображении слева показан пример фрагмента поля, а на изображении справа для этого же фрагмента показаны все возможные ходы для белой шашки: темно-зелеными крестиками помечены соседние клетки, в которые белая шашка может совершить ход (копирование), светло-оранжевыми — почти соседние клетки, в которые та же самая шашка может совершить ход (перенос).

Каждый раз, когда на какой-то клетке поля появляется шашка (в результате переноса или копирования), все шашки, которые находятся на соседних клетках, становятся того же цвета, что и эта шашка.

Если какой-то игрок не может сделать ход, то все оставшиеся пустые клетки (если таковые имеются) заполняются шашками соперника. По окончании игры подсчитываются шашки каждого из игроков. Победителем является тот, у кого на момент окончания игры на поле осталось больше

шашек. Если у обоих игроков на поле осталось одинаковое количество шашек, то результатом игры является ничья.

Паша является большим поклонником этой игры. Он решил написать программу, которая будет совершать ходы, чтобы играть с соперниками вместо самого Паши. С чего-то надо начинать создание программы-игрока. Паша решил, что неплохо было бы написать программу, которая умела бы определять на какое наибольшее число D можно увеличить количество своих шашек для заданного состояния поля. А для найденного наибольшего числа требуется также определить количество ходов T, которые приведут к увеличению на такое количество шашек. Для определенности будем считать, что шашками своего цвета являются белые шашки. Гарантируется, что хотя бы одна белая шашка на поле имеется. Гарантируется, что существует хотя бы один корректный ход.

Формат входного файла

Во входных данных содержится описание игрового поля. Описание состоит из 12 строк, каждая их которых содержит по 12 символов. Символ «.» (точка) обозначает свободную активную ячейку, символ «X» — неактивную ячейку, символ «B» — ячейку, занятую черной шашкой, а символ «W» — ячейку, занятую белой шашкой.

Формат выходного файла

В единственной строке выходных данных выведите два целых числа — D и T, разделенных одним пробелом.

Примеры

stdin	stdout
W.........B ....B ....B......B ...B.B.....B ....WWW..... .....XB..... ....W.B..... .... B.......... BB......... BBBBB......W	3 7

Задача D. Помогите архонту

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Архонт — высшее должностное лицо в древнегреческих полисах. Одному из мудрейших Афинских архонтов стало интересно, насколько надежно защищены границы его владений.

Границу владений этого архонта для простоты можно представить в виде прямой, вдоль которой расположены N сторожевых пунктов. Для удобства пронумеруем эти пункты слева направо последовательными целыми числами от 1 до N. Пункт с номером i имеет подземный ход в пункт A_i, который используется для передачи сообщений. Особенность этой системы передачи сообщений заключается в том, что $A_i \neq A_j$ при $i \neq j$ и $A_i \neq i$.

В случае ожидаемой атаки к некоторым сторожевым пунктам будет отправлен гонец, который сообщит о надвигающейся угрозе. После чего из каждого сторожевого пункта, которые оповещены об угрозе, по подземному ходу побежит гонец в другой сторожевой пункт и сообщит ему об угрозе. Далее из вновь оповещенных пунктов так же побегут гонцы и оповестят пункты, в которые ведут подземные ходы из их сторожевых пунктов. И так далее. Гонцы не побегут лишь в том случае, если они уже знали о надвигающейся угрозе к моменту, когда к ним прибежал другой гонец. Очевидно, что рано или поздно все забеги прекратятся. Афинские бегуны настолько суровы, что все забеги гарантированно прекратятся менее, чем за сутки.

Архонту стало известно, что враги готовы атаковать некоторый непрерывный отрезок сторожевых пунктов. Архонт не знает какой именно, но у него есть M предположений на этот счет. Предположение с номером i имеет вид (L_i, R_i), которое значит, что возможна атака на отрезок сторожевых пунктов от L_i до R_i.

Теперь архонт для каждого из своих предположений i хочет знать, какое минимальное количество гонцов ему потребуется, чтобы оповестить все сторожевые пункты на участке от L_i до R_i менее чем за сутки.

Ваша задача — написать программу, которая сможет решить точно такую же задачу.

Формат входного файла

Первая строка входных данных содержит единственное целое число N ($1 \leqslant N \leqslant 2 \cdot 10^5$) — количество сторожевых пунктов. Вторая строка содержит N различных целых чисел A_i ($1 \leqslant A_i \leqslant N$) — номер соседнего пункта для каждого из пунктов.

Третья строка входных данных содержит единственное целое число M ($1 \leqslant M \leqslant 10^5$) — количество предположений архонта. Далее следует M строк, каждая из которых описывает очередное предположение и содержит по два целых числа L_i и R_i, разделенных пробелом ($1 \leqslant L_i \leqslant R_i \leqslant N$).

Формат выходного файла

Для каждого из предположений на отдельной строке необходимо определить минимальное необходимое число гонцов в порядке, в котором эти предположения даны во входных данных.

Примеры

stdin	stdout
5 2 5 4 3 1 3 1 3 3 4 5 5	2 1 1

Задача E. Карты

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Скоро лето и многие собираются в отпуск. Программист Саша — не исключение. Да вот только выбрать, где же провести свой отпуск, является для Саши непосильной задачей. Саша решил поступить как настоящий программист. Он взял две квадратные карты одной и той же местности. Вторая карта строго меньше в размере, чем первая, и получена из нее путем пропорционального уменьшения в несколько раз. Саша положил вторую карту на первую так, чтобы вторая карта была полностью размещена на первой (так как Саша положил вторую карту довольно небрежно, она вполне может быть повернута относительно первой на некоторый угол).

Теперь Саша хочет найти такую точку на первой карте, которая на обеих картах указывает на одно и тоже географическое место, или определить, что такой точки нет. Для удобства представления положения второй карты Саша ввел систему координат так, чтобы левый нижний угол первой карты находился в точке $(0, 0)$, а правый верхний угол — в точке $(1, 1)$

Ваша задача — написать программу, которая поможет Саше в решении этой нелегкой задачи.

Формат входного файла

Единственная строка входных данных содержит четыре вещественных числа — координаты X_1, Y_1, X_2, Y_2 двух противоположных углов второй карты, размещенной на первой карте. Угол (X_1, Y_1) соответствует левому нижнему углу первой карты, угол (X_2, Y_2) — правому верхнему углу исходной карты. Все координаты даны не более, чем с тремя знаками после десятичной точки. Гарантируется, что заданный квадрат полностью лежит внутри квадрата, описывающего первую карту.

Формат выходного файла

В единственной строке выходных данных выведите два вещественных числа X_v и Y_v — координаты искомой точки. Если такой точки нет, то дважды выведите -1 (минус единицу). Координаты при выводе следует разделять одним пробелом. Ответ должен отличаться от правильного не более, чем на 10^{-6}.

Примеры

stdin	stdout
0.25 0.5 0.5 0.75	0.3333333333 0.6666666667
0.5 0.5 0.25 0.75	0.3529411765 0.5882352941
0.091 0.517 0.084 0.450	0.1020906483 0.4956000777

Задача F. Математическое ожидание

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Хорошо быть математиком! Даже самое обычное ожидание автобуса на остановке превращается в *математическое ожидание*. В чем же заключается это *математическое ожидание*? А заключается оно в том, что настоящий математик не просто ждет своего автобуса, а запоминает номера всех автобусов, которые подъезжают к остановке вместе с моментами времени, в которые это происходит.

После того, как математик садится в автобус, он начинает анализировать ту информацию, что он запомнил, пока стоял на остановке. В частности, каждый день математик выбирает лучший маршрут дня и худший маршрут дня.

Для того, чтобы определить лучший маршрут дня он выбирает такой маршрут, у которого наибольшее количество времени между приездами двух автобусов этого маршрута минимально. Если таких маршрутов несколько, то он выбирает тот, номер которого максимален.

Для того, чтобы определить худший маршрут дня он выбирает такой маршрут, у которого наименьшее количество времени между приездами двух автобусов этого маршрута максимально. Если таких маршрутов несколько, то он выбирает тот, номер которого минимален.

Стоит учесть, что все маршруты, автобусы которых за время *математического ожидания* подъехали к остановке только один раз, математик не учитывает при определении лучшего и худшего маршрутов дня.

Ваша задача — написать программу, которая по данным в памяти математика сможет определить лучший и худший маршруты дня.

Гарантируется, что есть хотя бы один маршрут, который будет учтен при определении лучшего и худшего маршрутов. Возможна ситуация, когда один и тот же маршрут станет в этот день и лучшим и худшим.

Формат входного файла

Первая строка входных данных содержит единственное число N — количество автобусов, которые подъезжали к остановке за время *математического ожидания* ($2 \leqslant N \leqslant 10^5$). Вторая строка содержит N целых чисел A_i, разделенных пробелом ($1 \leqslant A_i \leqslant 999$). Такая запись означает, что через i минут после того, как математик подошел к остановке, к ней подъехал автобус с номером A_i.

Формат выходного файла

В единственной строке выходных данных выведите два целых числа — номер лучшего маршрута дня и номер худшего маршрута дня, разделенные одним пробелом.

Примеры

stdin	stdout
7 26 23 4 4 23 32 26	4 26

Задача G. Слепое множество

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

У Паши есть свое мнение насчет того, как надо проводить свободное время. Сегодня он подготовил N карточек, на каждой из которых написал некоторое целое неотрицательное число A_i, после чего он положил все эти карточки в мешок.

Также у Паши есть изначально пустое множество целых чисел S.

До тех пор, пока в мешке есть хотя бы одна карточка, Паша будет тянуть карточки по одной из мешка и добавлять написанное на карточке число во множество S только в том случае, если после добавления этого числа множество S останется *понятным*. Все порядки вытягивания карточек можно считать равновероятными.

Множество называется *понятным*, если в нем нету двух таких чисел, что для некоторого разряда десятичной записи в обоих этих числах находится цифра отличная от нуля. Например, множества $\{0, 10, 203, 5000\}$, $\{1337\}$ и $\{5, 30\}$ являются *понятными*, а множества $\{5, 11, 300\}$ и $\{5, 6\}$ — не являются.

После того, как Паша достал все карточки из мешка, он подсчитывает сумму элементов полученного множества S.

Паше стало интересно, чему равняется значение математического ожидания суммы элементов множества S. Сам он, конечно, не может посчитать это значение.

Ваша задача — попытаться написать программу, которая посчитает искомое значение математического ожидания.

Формат входного файла

Первая строка входных данных содержит целое число N — количество карточек в мешке ($1 \leqslant N \leqslant 2 \cdot 10^5$).

Во второй строке входных данных содержится N целых чисел A_i — числа, которые были записаны на карточках ($0 \leqslant A_i \leqslant 10^{15}$).

Формат выходного файла

В единственной строке выходных данных выведите вещественное значение искомого математического ожидания.

Относительная или абсолютная погрешность этого значения не должна превышать 10^{-6}.

Примеры

stdin	stdout
8 1002 101 322 9 100 666 2000 2013	2066.581250000000200

Задача H. Результаты соревнований

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

На сайте посвященном олимпиадному программированию в Томском политехническом университете периодически проводится серия коротких соревнований — так называемые ABC раунды. Первый такой раунд был проведен в далеком 2009 году. Каждый раунд состоит из трех задач, на решение которых участникам отводится 120 минут.

За каждую решенную задачу участнику начисляется штрафное время, которое является целым числом и измеряется в минутах. Если участник сдал задачу на K-ой минуте, то ему начисляется K минут штрафного времени. Очевидно, K может принимать значения от 1 до 120. При подведении итогов более высокое место занимает участник, который решил наибольшее количество задач. При равенстве количества решенных задач, выше оказывается участник, у которого суммарное штрафное время за все решенные задачи меньше. Стоит отметить, что в настоящих правилах ABC раундов еще предусматривается дополнительное штрафное время за неудачные отправки, но в условиях данной задачи это не имеет значения.

Как-то раз Илья рассматривал результаты недавнего ABC раунда. Он особенно внимательно следит за выступлением Димы и Максима. Он выписал на листок в хронологическом порядке события в виде пар [*участник решивший задачу, решенная им задача*] и штрафное время, которое к концу раунда было в таблице у Димы и Максима. Уже на следующий день сайт с таблицей результатов перестал быть доступен по техническим причинам.

Илье стало интересно, сколько существует способов распределить записанные им события по минутам соревнования так, чтобы получились записанные им значения штрафного времени. Он прекрасно помнил, что никакие два события не произошли в одну и ту же минуту. Также Илья помнит, что Дима и Максим не делали неверных попыток.

Ваша задача — написать программу, которая найдет искомое количество способов распределить события по минутам соревнования.

Формат входного файла

Первая строка входных данных содержит два целых числа T_D и T_M — штрафное время в минутах Димы и Максима соответственно ($0 \leqslant T_D, T_M \leqslant 180$). Вторая строка входных данных содержит описание всех событий, которые записал Илья. Каждое описание задается двумя символами. Первый символ всегда равен «`D`», если задачу решил Дима, или «`M`», если задачу решил Максим. Второй символ всегда равен «`A`», «`B`» или «`C`» — литера задачи, которую сдал участник. Все события разделяются ровно одним пробелом. Никакое событие не упоминается дважды. Гарантируется, что Илья записал хотя бы одно событие.

Формат выходного файла

В единственной строке выходных данных выведите искомое число способов.

Примеры

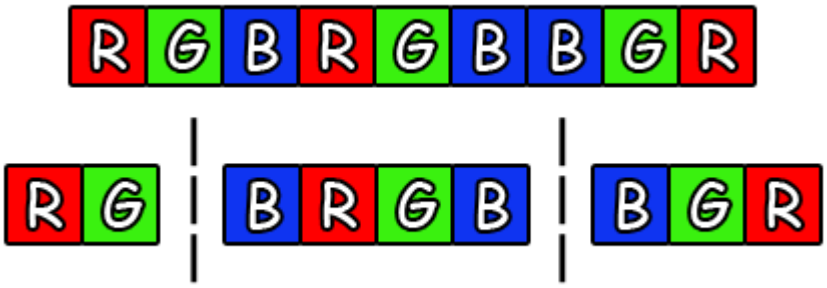
stdin	stdout
6 3 DA MA DB	2
23 125 DA MA MB DB MC	715

Задача I. Игра с лентой

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Имеется лента. Лента разбита на N участков одинаковой длины, каждый из которых имеет определенный цвет.

Два игрока играют в следующую игру. Игроки делают ходы по очереди. В свой ход каждый из игроков может вырезать из ленты некоторую часть, состоящую не менее, чем из двух участков. Вырезать можно только часть ленты, у которой первый и последний участок имеют одинаковый цвет. Более того, лента, которую вырезает участник, не должна содержать других участков этого цвета. Вырезать часть ленты необходимо обязательно двумя разрезами, то есть так, чтобы после того, как лента вырезана, образовалось три ленты. Причем все три ленты, которые получились после хода игрока, остаются в игре.



На каждом ходу игрок может выбрать любую ленту из тех, что участвуют в игре, и вырезать любую часть этой ленты, удовлетворяющую описанным требованиям. Тот, кто первый не может сделать ход, проигрывает.

Для того, чтобы оценить свои шансы игроки хотят знать, кто победит при оптимальной игре обоих игроков для указанной ленты.

Ваша задача — написать программу, которая для заданной ленты определит победителя.

Формат входного файла

Единственная строка входных данных содержит строку S, состоящую из N символов ($2 \leqslant N \leqslant 200$). Строка S задает исходную ленту, которая состоит из N сегментов. Каждый из символов задает очередной участок ленты. Символ на i-ой позиции задает цвет i-ого по счету слева участка ленты. Каждый из символов — заглавная буква латинского алфавита. Каждая буква соответствует ровно одному цвету.

Формат выходного файла

Единственная строка выходных данных должна содержать «`First`» (без кавычек), если победит игрок, который ходит первым, или же «`Second`» (без кавычек) — если победит игрок, который ходит вторым.

Примеры

stdin	stdout
RGBRGBBGR	First
TOMSKPROGRAMMINGCUP	Second

Задача J. Слон и Ладья

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Как-то раз Паша и Илья играли в шахматы. Паша проиграл и, чтобы хоть как-то отомстить Илье, украл у него две шахматные фигуры — белого слона и черную ладью.

Паша пришел домой, начертил на огромном листе бумаги шахматное поле размера $N \times N$. И поставил на него обе фигуры. После чего он заметил, что одна из фигур находится под боем другой фигуры. Пашу заинтересовал вопрос: сколько существует способов расставить белого шахматного слона и черную шахматную ладью на доске размера $N \times N$ так, чтобы ни одна из фигур не находилась под боем.

Белый слон находится под боем черной ладьи, если эти две фигуры находятся в одной строке или одном столбце. Черная ладья находится под боем белого слона, если эти две фигуры находятся на одной диагонали.

Ваша задача — написать программу, которая поможет для заданного размера N определить искомое количество способов.

Формат входного файла

Единственная строка входных данных содержит единственное целое число N — размер шахматного поля ($2 \leqslant N \leqslant 200$).

Формат выходного файла

Единственная строка выходных данных должна содержать число способов осуществить корректную расстановку.

Примеры

stdin	stdout
3	16

Note

Стоит обратить внимание на то, что цвет шахматных фигур не накладывает никаких ограничений на цвет клеток, на которых можно эти фигуры располагать.

Задача K. Создание квадрата

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

У Паши есть N палочек, каждая из которых имеет определенную целую длину l_i. Паша хочет выложить с помощью этих палочек квадрат наибольшей возможной площади. Он хочет, чтобы стороны квадрата состояли ровно из одной палочки каждая. Для этого ему разрешается распиливать палочки произвольным образом любое количество раз, но только так, чтобы длины палочек, получившихся после распиливания, были представлены целыми числами. Так, например, из палочки длины 13 можно получить четыре палочки длины 3 и одну палочку длины 1. Склеивать палочки Паше не разрешается.

Паше интересно знать, какую наибольшую сторону может иметь квадрат, который он сможет выложить из имеющегося у него набора палочек. Ему не хватает ни ума, ни фантазии, чтобы решить такую задачу.

Ваша задача — написать программу, которая за Пашу найдет искомую наибольшую сторону квадрата.

Формат входного файла

Первая строка входных данных содержит единственное целое число N — количество палочек у Паши ($1 \leqslant N \leqslant 2 \cdot 10^3$). Вторая строка содержит N целых чисел l_i — длины палочек ($1 \leqslant l_i \leqslant 10^6$). Гарантируется, что сумма всех l_i строго больше трех.

Формат выходного файла

Единственная строка выходных данных должна содержать единственное целое число — искомую наибольшую возможную длину стороны квадрата.

Примеры

stdin	stdout
5 6 4 3 2 1	3

Note

В приведенном в примере наборе входных данных необходимо распилить первую палочку на две палочки длины 3, вторую — на палочки длины 3 и 1. После чего в наборе окажется четыре палочки длины 3, из которых Паша сможет собрать квадрат со стороной длины 3. Квадрат с большей стороной Паша составить не сможет.