Задача 1. ICPC Contest Resolver

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

Команда Noname State University (NSU) соревнуется в финале чемпионата мира по программированию ACM ICPC. Прошло четыре часа с начала соревнования, рейтинг только что заморозили, а дела у нашей команды не очень. К счастью, на помощь несчастным программистам пришёл сам Чак Норрис. Кажется, с его помощью команда может даже стать чемпионом мира! Теперь команде хочется по возможности сделать свою победу наиболее зрелищной на церемонии раскрытия результатов.

Несколько фактов о Чаке Норрисе:

- Чак Норрис решил все задачи задолго до того, как они были придуманы.
- Чак Норрис пишет задачи настолько быстро, насколько ему позволяет оперативная память компьютера.
- В каждую минуту Чак Норрис может сделать любое количество посылок по каждой задаче. Из принципа он согласен посылать решения только за команду NSU.
- Чак Норрис может добиться любого возможного вердикта по каждой своей посылке, однако из жалости к жюри он решил ограничиться вердиктами “ACCEPTED” и “Wrong Answer”.
- Чак Норрис может предсказывать будущее, поэтому он знает всё обо всех посылках других команд на протяжении всего тура, в том числе после заморозки.

Соревнование длится 300 минут. Всего имеется K задач, пронумерованных числами от 1 до K в возрастающем порядке. Любая команда в течение тура может отправлять на проверку решение любой задачи. Система тестирования проверяет посылки участников в порядке их поступления и возвращает для каждой из них вердикт. Посылка с вердиктом “ACCEPTED” считается успешной, а все остальные — неуспешными. Задача считается решённой командой, если по ней есть хотя бы одна успешная посылка от этой команды. Штрафное время по задаче равно количеству полных минут от начала тура до первой успешной посылки плюс количество посылок до первой успешной посылки, умноженное на 20. При подсчёте штрафного времени по задаче учитываются посылки только по этой задаче.

Команды в рейтинге сортируются по убыванию количества решённых задач. Команды с равным количеством задач ранжируются по общему штрафному времени по возрастанию. Общее штрафное время равно сумме штрафного времени по всем решённым задачам. Если у команд совпадает количество решённых задач и штрафное время, то они сортируются в порядке увеличения номера команды.

На протяжении первых 240 минут соревнования доступен общий рейтинг на текущий момент. Затем рейтинг «замораживается», то есть перестаёт обновляться. На церемонии закрытия результаты посылок за последний час раскрываются постепенно при помощи программы ICPC Contest Resolver.

Contest Resolver работает следующим образом. Изначально он показывает рейтинг на момент заморозки соревнования. Считается, что все посылки, произведённые после этого, ещё не проверены. Непроверенные посылки никак не влияют на рейтинг, как будто их нет вовсе. Пока есть хотя бы одна непроверенная посылка, происходит пересчет, который заключается в поочерёдном выполняются следующих шагов.

- В рейтинге выбирается команда с наибольшим местом, у которой есть хотя бы одна непроверенная посылка.

- Среди задач выбирается задача с наименьшим номером, по которой у этой команды есть непроверенная посылка.
- Все непроверенные посылки этой команды по этой задаче проверяются в хронологическом порядке.
- После этого рейтинг обновляется с учётом проверенных посылок. В результате команда может совершить прыжок вверх, то есть занять более высокое положение в рейтинге.

Величиной прыжка называется разность между местом до выполнения пересчета и местом после выполнения.

Назовём *зрелищностью выступления команды* сумму квадратов величин всех прыжков команды в процессе работы Contest Resolver. Следует заметить, что увеличение места команды в результате проверки успешных решений других команд прыжком не считается. Требуется найти оптимальное поведение команды (вместе с Чаком Норрисом, конечно) после заморозки. Оптимальное поведение определяется следующими условиями:

1. Команда должна стать чемпионом, то есть занять первое место в финальном рейтинге, если это возможно.
2. Зрелищность выступления должна быть наибольшей при условии выполнения первого пункта.

Если вариантов оптимального поведения несколько, разрешается вывести любой.

Чак Норрис **не** умеет менять прошлое, поэтому количество полных минут от начала соревнования до любой дополнительной посылки должно быть в диапазоне от 240 до 299. Будем считать, что невозможно отправить задачу в точности через целое количество минут после начала соревнования. Так, например, нельзя получить за сданную с первой попытки задачу штрафное время 300.

Формат входного файла

В первой строке входного файла записаны через пробел три целых числа K, N и M, где K — количество задач в соревновании, N — количество участвующих команд и M — количество посылок, записанных в данном файле ($1 \le K \le 10$, $1 \le N \le 10^3$, $0 \le M \le 10^5$).

В каждой из следующих M строк описано по одной посылке. Посылка описывается в формате t a p r, где t — количество полных минут от начала соревнования до момента посылки, a — номер команды, p — номер задачи и r — вердикт проверки ($0 \le t \le 299$, $1 \le a \le N$, $1 \le p \le K$). Числа t, a, p — целые. Вердикт r задаётся одной буквой: '`A`' для успешной посылки и '`W`' для неуспешной. Посылки даны в порядке их поступления в систему тестирования.

Номер нашей команды NSU равен 1. Все заданные посылки нашей команды выполнены до заморозки, то есть если для посылки верно $a = 1$, то также верно $t \le 239$.

Гарантируется, что у любой команды, кроме NSU, общее штрафное время в финальном рейтинге не превосходит 2 500.

Формат выходного файла

Если команда NSU не может стать чемпионом, то в единственную строку выходного файла необходимо вывести слово "`Losers`".

В противном случае в первую строку нужно вывести слово "`Champions`", а во вторую строку записать одно целое число — зрелищность выступления команды при оптимальном поведении.

Далее нужно вывести оптимальное поведение. В третьей строке должно быть записано целое число Q — количество посылок после заморозки. В каждой из следующих Q строк должна быть выведена информация о соответствующей посылке. Формат описания посылки

должен быть таким же, как во входных данных. Все посылки требуется выводить в порядке их поступления в систему тестирования.

Общее количество посылок Q не должно превышать 10^4. Гарантируется, что если команда NSU может стать чемпионом, то существует оптимальное поведение, удовлетворяющее этому ограничению.

Пример

input.txt	output.txt
8 5 16 0 1 1 W 30 1 1 A 30 1 1 W 30 1 1 W 30 1 1 A 40 2 1 A 50 3 1 A 60 4 1 A 120 5 4 W 150 2 2 W 170 2 2 A 239 3 2 A 240 4 3 A 260 5 3 W 270 5 4 A 290 4 4 A	Champions 10 2 250 1 2 A 290 1 3 A

Комментарий

В примере на момент заморозки монитор выглядел так:

M	Команда	1	2	3	4	5	=	Штраф
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	?	?	.	.	1	50
4	#4	+	.	?	?	.	1	60
5	#5	.	.	?	?1	.	0	0

Сначала были проверены посылки команды номер 5 по задачам 3 (неуспешная) и 4 (успешная). Так как пятая команда осталась на последнем месте, то положение команд не изменилось:

M	Команда	1	2	3	4	5	=	Штраф
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	?	?	.	.	1	50
4	#4	+	.	?	?	.	1	60
5	#5	.	.	-1	+1	.	1	290

После этого была проверена посылка команды номер 4 по задаче 3, которая оказалась успешной, и команда поднялась на одно место вверх:

M	Команда	1	2	3	4	5	=	Штраф
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#4	+	.	+	?	.	2	300
4	#1	+1	?	?	.	.	1	50
5	#5	.	.	-1	+1	.	1	290

Вслед за этим была принята задача 2 у команды NSU, которая в результате поднялась с 4 места на 3, а зрелищность ее выступления увеличилась на 1:

M	Команда	1	2	3	4	5	=	Штраф
1	#2	+	+1	.	.	.	2	230
2	#3	+	+	.	.	.	2	289
3	#1	+1	+	?	.	.	2	300
4	#4	+	.	+	?	.	2	300
5	#5	.	.	-1	+1	.	1	290

В результате успешной сдачи задачи 4 команда номер 4 поднялась на первое место:

M	Команда	1	2	3	4	5	=	Штраф
1	#4	+	.	+	+	.	3	590
2	#2	+	+1	.	.	.	2	230
3	#3	+	+	.	.	.	2	289
3	#1	+1	+	?	.	.	2	300
5	#5	.	.	-1	+1	.	1	290

В конце была проверена успешная посылка команды NSU по задаче 3, команда поднялась с 4 места на первое, и в результате зрелищность ее выступления увеличилась на 9:

M	Команда	1	2	3	4	5	=	Штраф
1	#1	+1	+	+	.	.	3	590
2	#4	+	.	+	+	.	3	590
3	#2	+	+1	.	.	.	2	230
4	#3	+	+	.	.	.	2	289
5	#5	.	.	-1	+1	.	1	290

Задача 2. Рога и копыта

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

Молочные братья Паниковский и Балаганов после неудачного покушения на подпольного миллионера Корейко с горя заперлись в конторе и играют в очень странную игру. У одного в распоряжении полная комната рогов, а у другого — копыт. Они берут все возможные пары (*рог*, *копыто*). Чей предмет оказался дороже, тот и побеждает.

Их интересует актуальный вопрос: кто будет чаще побеждать? Правда, душевное здоровье у них подорвано, поэтому с этой задачей они могут не справиться. Помогите им, а уж они вас отблагодарят — и рогами, и копытами.

Формат входного файла

В первой строке входного файла содержатся два числа n и m, где n — количество сортов рогов, m — количество сортов копыт ($1 \le n, m \le 10^5$).

Далее в n строках описываются рога, по одному сорту в строке. Каждый сорт задается двумя целыми положительными числами, первое из которых — это цена одного рога данного сорта, а второе — запасы рогов этого сорта (количество штук).

Далее в m строках идут в таком же формате описания сортов копыт.

Цена экземпляра любого сорта рогов или копыт не превосходит 10^9.

Количество экземпляров рогов и копыт каждого сорта не превышает 10^4.

Формат выходного файла

В выходной файл необходимо вывести в одной строке через пробел три целых числа. Первое число — это количество пар рогов и копыт, где стоимость рога больше стоимости копыта, второе — количество пар с равной стоимостью рога и копыта, а третье — количество пар, в которых стоимость рога меньше стоимости копыта.

Пример

input.txt	output.txt
3 2 1 2 2 3 4 1 2 3 3 1	4 9 11

Задача 3. Строка

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

В НИИ Данных Строк изучается новый способ построения бесконечных строк.

Первоначально имеется пустая строка $S_0^{(k)} = \varepsilon$. Каждая следующая версия строки получается по описанной ниже схеме. Текущая версия строки повторяется k раз, а между каждыми двумя соседними повторениями вставляется произвольный символ. Для построения всех версий используется одинаковое число k, однако вставляемые символы могут быть различными. При таком построении в пределе получается бесконечная строка $S_\infty^{(k)}$.

В качестве примера можно привести серию строк с $k = 3$:

$$
\begin{aligned}
S_0^{(3)} &= \varepsilon(\text{пустая}) \\
S_1^{(3)} &= \mathbf{rt} && (S_1^{(3)} = \boxed{\varepsilon}\mathbf{r}\boxed{\varepsilon}\mathbf{t}\boxed{\varepsilon}) \\
S_2^{(3)} &= \boxed{rt}\mathbf{x}\boxed{rt}\mathbf{r}\boxed{rt} \\
S_3^{(3)} &= \boxed{rtxrtrrt}\mathbf{a}\boxed{rtxrtrrt}\mathbf{r}\boxed{rtxrtrrt} \\
&\dots \\
S_\infty^{(3)} &= rtxrtrrtartxrtrrtrrtxrtrrtzrtxrtrrtartxrt\dots
\end{aligned}
$$

Дана некоторая строка A длины n, которая является началом $S_\infty^{(k)}$. Требуется найти минимальное k, для которого это возможно. Иными словами, требуется найти минимальное k, для которого можно построить строку $S_\infty^{(k)}$ описанным выше образом, так что она будет начинаться строкой A.

Формат входного файла

Во входном файле содержится несколько тестов. В первой строке файла записано одно целое положительное число T — количество тестов. Далее в файле идут T строк, каждая строка описывает один тест, эти строки непусты и состоят лишь из строчных латинских букв.

Количество тестов не превосходит 10^5. Суммарная длина всех тестов в файле не превосходит 10^6.

Формат выходного файла

В выходном файле должно быть ровно T строк. В каждой строке должно быть записано минимальное значение k для соответствующего теста.

Пример

input.txt	output.txt
2 abacabab rtxrtrrtartxrtrrtrrtxrt	2 3

Задача 4. Книги

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 МиБ

Группа спортивного программирования Неизвестного государственного университета (НГУ) готовится к новому виду соревнований. В этом соревновании от университета может участвовать только одна команда, состоящая из двух человек. Также есть официальный список из N книг, которые участники должны освоить для успешного выступления. До соревнования остаётся не так много времени, и не каждый студент способен освоить все книги вовремя.

Каждый студент оценил для каждого множества книг, успеет ли он освоить его до начала соревнования. Всего в НГУ тренируется M студентов. Тренер подготовил список из K потенциальных команд. Чтобы выбрать единственную команду для участия в соревновании, он хочет узнать для каждой команды её возможности по освоению книг.

Будем считать, что команда освоила книгу, если хотя бы один из её членов освоил эту книгу. Требуется определить для каждой потенциальной команды и для каждого множества книг, могут ли участники команды спланировать подготовку таким образом, чтобы к началу соревнования команда освоила это множество книг.

Формат входного файла

В первой строке входного файла даны три целых числа: N — количество книг, M — количество студентов и K — количество потенциальных команд ($1 \le N \le 21$, $2 \le M \le 6$, $1 \le K \le 6$). В данной задаче всюду используется нумерация, начиная с нуля.

В i-ой из следующих M строк заданы возможности i-ого студента. Возможности студента описываются строкой S_i длины 2^N из нулей и единиц. В k-ой позиции строки записано, успевает ли студент освоить множество книг, битовая маска которого равна k. Данное правило более подробно расписано ниже.

Пронумеруем все книги от 0 до $N-1$. Рассмотрим произвольное множество книг X. Определим последовательность $a_0, a_1, a_2, \ldots, a_{N-1}$, такую что $a_j = 1$, если j-я книга входит в множество X, и $a_j = 0$ иначе. Назовём битовой маской множества X число

$$k = \sum_{j=0}^{N-1} a_j 2^j.$$

Тогда i-ый студент успевает освоить множество книг X тогда и только тогда, когда k-ый символ строки S_i равен единице.

В t-ой из последующих K строк описана t-я потенциальная команда. Команда задаётся двумя целыми числами a_t и b_t — номерами студентов, которые в ней состоят. ($0 \le a_t, b_t < M$, $a_t \ne b_t$, $t = 0 \ldots K-1$).

Относительно возможностей каждого студента гарантируется следующее. Во-первых, если он может освоить некоторое множество книг, то он может освоить и любое его подмножество. Во-вторых, студент всегда успевает освоить пустое множество книг, то есть в строке его возможностей S_i нулевой символ всегда равен единице.

Формат выходного файла

В выходной файл необходимо вывести K строк из нулей и единиц, по 2^N символов в каждой. В t-ой строке должны быть записаны возможности t-ой команды в том же формате,

в котором задаются возможности студентов во входе. То есть k-ый символ t-ой строки должен быть равен 1 тогда и только тогда, когда команда под номером t может к началу соревнования освоить множество книг X, битовая маска которого равна k. В противном случае он равен нулю.

Пример

input.txt	output.txt
3 5 3 11111010 11000000 11001000 11101000 11101000 0 1 1 2 3 4	11111111 11001100 11111110

Комментарий

Нулевой студент может освоить одно из множеств: $\emptyset$, $\{0\}$, $\{1\}$, $\{2\}$, $\{0,1\}$, $\{1,2\}$. Первый студент может освоить только книгу под номером ноль (либо ничего). Второй студент может освоить либо книгу под номером ноль, либо книгу под номером два (либо ничего). Третий и четвёртый студенты могут освоить одну произвольную книгу (либо ничего).

Нулевая команда может вместе освоить все книги, если нулевой студент освоил книги $\{1,2\}$, а первый — книгу 0. Нетрудно проверить, что любое другое множество книг эта команда также может освоить.

Первая команда может освоить книги 0 и 2, если студенты будут осваивать разные книги. Также команда может освоить любое подмножество множества $\{0,2\}$.

Во второй команде каждый участник может освоить не более одной книги. Значит вместе команда может освоить любое множество из двух или менее книг.

Задача 5. Кровать

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Где-то недалеко от нас, на планете Нибиру, один из самых перспективных студентов Василиус получил повышенную стипендию в престижнейшем Нибирийском государственном университете (НГУ) и решил купить себе кровать в общежитие. Василиус хочет купить как можно большую кровать. С его точки зрения, с одной стороны, такая кровать должна иметь максимальный объем, но с другой стороны, размеры будущей кровати ограничены — Василиус хочет оставить место в комнате под новые покупки.

В магазине могут изготовить на заказ любую кровать, имеющую форму обычного земного прямоугольного параллелепипеда. Но не всякую кровать можно доставить из магазина в комнату общежития. Её придется проносить сквозь порталы, которые напоминают по виду наши обычные двери прямоугольной формы. Плоскости порталов перпендикулярны плоскости пола. В результате прохождения сквозь портал можно оказаться либо возле каких-то других порталов, либо в каком-то помещении. Расстояние между порталами намного больше размеров комнаты Василиуса.

Помогите найти оптимальный размер кровати, которую Василиус сможет пронести к себе в комнату.

Предполагается, что кровать из магазина будут нести так, чтобы одна её фиксированная грань всегда оставалась параллельной полу, а при прохождении через порталы другая фиксированная грань была параллельна плоскости портала. При этом вращать кровать после того, как её начнут нести сквозь порталы до комнаты, не будут.

Формат входного файла

В первой строке входного файла содержатся три целых числа a, b и c — ограничения на размеры кровати. Один из параметров кровати (длина, ширина, высота) не должен превосходить a, другой b, а третий c ($1 \le a, b, c \le 500$).

Следующая строка содержит одно число n — количество порталов на планете ($2 \le n \le 500$). Все порталы пронумерованы числами от 1 до n. Считаем, что портал в магазине имеет номер 1, а портал, который приводит в комнату Василиуса, имеет номер n.

Далее в n строках входного файла описываются порталы Нибиру в порядке возрастания их номеров, по одному на строке. Описание портала начинается с трех, записанных через пробел, целых чисел w, h и k, первые два из которых задают ширину и высоту портала, третье — количество порталов, к которым можно попасть напрямую, пройдя через описываемый портал ($1 \le w, h \le 300, 0 \le k < n$). Далее в этой строке через пробел записано k чисел — номера порталов, доступных из данного. Все эти порталы различны и отличаются от описываемого портала.

Формат выходного файла

В выходной файл необходимо в произвольном порядке вывести через пробел три целых положительных числа — высоту, ширину и длину максимальной по объему кровати, которую Василиус может пронести к себе в комнату из магазина. Если вариантов размеров с максимальным объемом несколько, то выведите любой из них. Гарантируется, что решение существует.

Примеры

input.txt	output.txt
300 300 300 4 300 300 1 2 100 200 1 4 300 200 1 1 300 300 2 3 1	100 200 300
300 300 300 4 300 300 2 2 3 100 200 1 4 300 200 1 4 300 300 1 1	200 300 300

Задача 6. Antichamber (Division 1 Only!)

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 МиБ

В компьютерной игре “Antichamber” у игрока есть специальный инструмент под названием “Brick Tool”. В данной задаче требуется смоделировать его работу в упрощённом двумерном варианте.

Действие происходит на бесконечном во все стороны клеточном поле. Каждая клетка может быть либо белой, либо чёрной. Две клетки считаются *смежными*, если они имеют общую сторону. *Компонентой* называется максимальный связный набор чёрных клеток. Размер компоненты определяется как количество клеток в ней.

Инструмент позволяет игроку перекрасить любую клетку в противоположный цвет. Сразу после этого выполняются проверки, в результате которых может измениться состояние поля.

Если после перекрашивания клетки в белый цвет увеличилось количество компонент, то это значит, что некоторая компонента распалась на несколько компонент-частей. В таком случае вновь образовавшиеся компоненты-части удаляются, за исключением, возможно, наибольшей из них по размеру. Наибольшая по размеру компонента не удаляется лишь в том случае, если её размер строго больше суммы размеров всех остальных компонент-частей. Удаление компоненты означает перекрашивание всех её клеток в белый цвет.

Затем независимо от цвета перекрашиваемой клетки устраняются «дыры» в компонентах: если какая-либо белая клетка находится внутри какого-либо цикла из чёрных клеток, то она перекрашивается в чёрный цвет. В цикле каждые две соседние клетки должны быть смежными.

Изначально все клетки белые. Дана последовательность операций. Каждая операция — либо перекрашивание клетки, либо запрос. Требуется выполнить операции в заданном порядке.

Бывает два типа запросов. Запрос типа `Comp`: верно ли, что заданные две клетки чёрные и лежат в одной компоненте? Запрос типа `Size`: определить размер компоненты, в которой лежит заданная клетка. Если эта клетка белая, то нужно вывести ноль в качестве ответа.

Формат входного файла

В первой строке входного файла записано целое число N — количество перекрашиваний и запросов ($1 \le N \le 10^5$). Каждая из следующих N строк содержит описание одной операции.

Форматы операций представлены ниже:

- “`Draw` x y” — перекрасить клетку с координатами (x, y) в противоположный цвет;
- “`Size` x y” — найти размер компоненты, содержащей клетку с координатами (x, y);
- “`Comp` x_1 y_1 x_2 y_2” — определить, лежат ли клетки с координатами (x_1, y_1) и (x_2, y_2) в одной компоненте.

Все координаты целые положительные и не превосходят 10^6.

Формат выходного файла

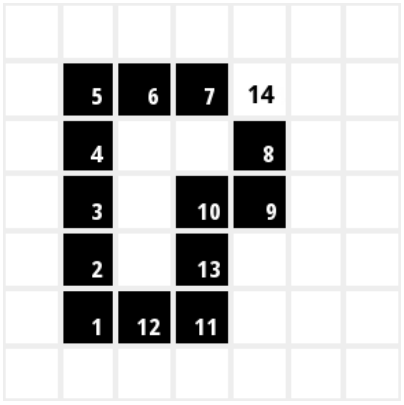
В выходной файл необходимо вывести ответы на запросы в порядке их следования во входном файле, каждый ответ в отдельной строке. Ответ на запрос типа `Size` должен быть неотрицательным целым числом. Ответ на запрос типа `Comp` должен быть “`YES`” или “`NO`”.

Пример

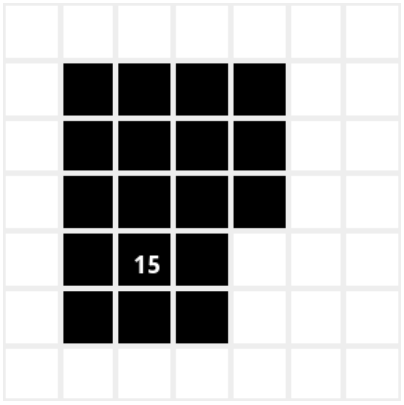
input.txt	output.txt
38 Draw 2 2 Draw 2 3 Draw 2 4 Draw 2 5 Draw 2 6 Draw 3 6 Draw 4 6 Draw 5 5 Draw 5 4 Draw 4 4 Draw 4 2 Size 4 2 Size 2 4 Size 4 4 Size 3 3 Comp 2 2 2 4 Comp 2 2 4 4 Comp 3 3 4 4 Draw 3 2 Draw 4 3 Size 2 2 Draw 5 6 Size 2 2 Draw 3 3 Size 3 3 Comp 3 3 4 4 Draw 2 4 Draw 3 4 Draw 4 4 Size 3 2 Size 3 6 Draw 4 7 Draw 3 5 Size 2 5 Draw 4 6 Size 2 5 Size 4 5 Size 4 7	1 7 3 0 YES NO NO 13 18 18 YES 0 9 9 0 0 0

Комментарий

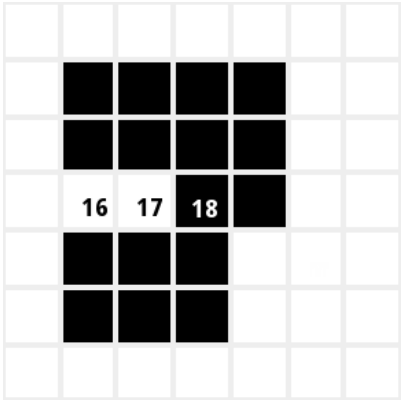
Числа на представленных иллюстрациях обозначают порядковый номер применения инструмента Brick tool для соответствующей клетки.



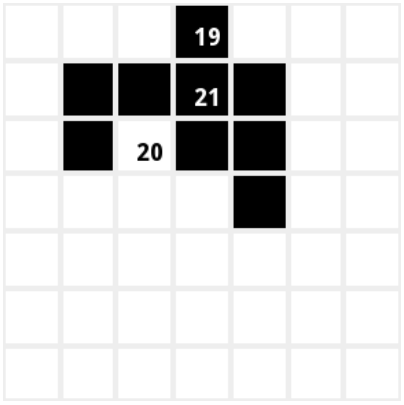
Состояние, в котором находится поле при четырнадцатом применении Brick tool показано на иллюстрации.



Пятнадцатое применение не изменяет состояния поля.



Состояние поля при восемнадцатом применении Brick tool.



После двадцать первого применения Brick tool все клетки поля станут белыми.

Задача 7. Хорёк в курятнике

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

Во время экспедиции к одной из дальних звёзд группа свободного поиска сделала очередную находку, по всей видимости, связанную с цивилизацией Странников. Находка представляла собой механизм неизвестного назначения, главным элементом пульта управления которого было прямоугольное клеточное поле размером $M \times N$, некоторые клетки которого были выколоты, а некоторые — окрашены в один из 26 различных цветов. Также рядом с пультом была обнаружена коробка с большим количеством прямоугольных пластинок размером 2×1 клетки, напоминающих земные костяшки домино. Размер клетки пластинки в точности соответствовал размеру клетки на пульте.

Известный специалист по разгадке намерений Странников Рудольф Сикорски предположил, что для активации механизма надо выложить пластины на поле так, чтобы ни одна из выколотых клеток не была бы закрыта, а из клеток каждого цвета была бы закрыта ровно одна. Незакрашенные и невыколотые клетки должны быть закрыты полностью.

Бывшему прогрессору Льву Абалкину удалось выкрасть коробку с пластинами. Сейчас он должен разместить пластины на поле так, чтобы механизм активировался. Помогите ему сделать это, пока Рудольф Сикорски не обнаружил пропажу.

Формат входного файла

В первой строке входного файла записаны через пробел два целых числа M и N — размеры поля ($1 \le M, N \le 100$).

Далее задаётся само поле: M строк, каждая из которых содержит по N символов — описаний клеток.

Выколотой клетке соответствует символ ‘.’ (ASCII 46), клетке одного из 26 цветов — заглавная латинская буква от ‘`A`’ до ‘`Z`’, при этом клеткам разных цветов соответствуют разные буквы, а одинаковых — одинаковые, остальным клеткам — символ ‘`*`’(ASCII 42).

Окрашенные клетки могут встречаться во входном файле в любом наборе. Гарантируется, что клетки, окрашенные одним цветом, имеют одинаковую четность. Четность клетки — это четность суммы ее координат.

Гарантируется, что существует хотя бы одна невыколотая клетка, окрашенная или нет.

Формат выходного файла

В первую строку выходного файла нужно вывести слово “`No`”, если решения не существует.

В противном случае нужно вывести слово “`Yes`”. В следующих строках требуется вывести раскладку пластинок: M строк по N символов в каждой.

Символ ’`.`’(ASCII 46) обозначает выколотую клетку поля.

Каждая пластина занимает две клетки, таким образом каждая покрытая клетка покрыта половинкой пластины: правой, левой, верхней либо нижней.

Верхняя часть пластины изображается символом ’`^`’(ASCII 94).

Нижняя часть пластины изображается символом ’`v`’(ASCII 118).

Левая часть пластины изображается символом ’`<`’(ASCII 60).

Правая часть пластины изображается символом ’`>`’(ASCII 62).

Те клетки поля, которые не были закрыты пластинами, обозначаются символом ’`@`’ (ASCII 64).

Понятия «верх», «низ», «лево» и «право» относятся к представлению выходного файла на экране, а не к самому полю.

Пример

input.txt	output.txt
4 8 ...***.. ..A.A*.. .B***.*. ******B.	Yes ...<>^.. ..^.@v.. .@v<>.^. <><><>v.

Задача 8. Колонизация

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 МиБ

В бутылке с йогуртом плавает N бактерий. Существовать в одиночестве бактериям скучно, поэтому они объединяются в колонии. Этот процесс происходит следующим образом. Первоначально каждая бактерия представляет собой колонию, состоящую из одной такой бактерии. Затем, пока в йогурте находится не менее двух колоний, на каждом шаге выбираются две колонии, которые находятся ближе всего друг к другу, и объединяются.

Для каждой пары бактерий u и v известно расстояние между ними $d_{u,v}$. Расстояние для колоний G и H вычисляется как среднее арифметическое расстояний между всеми парами бактерий:

$$\frac{1}{|G| \cdot |H|} \sum_{u \in G} \sum_{v \in H} d_{u,v}$$

Бактерии не очень придирчивы к точности, на то они и одноклеточные. Поэтому, если существует несколько пар колоний, расстояние между которыми отличается от минимального не более, чем на 10^{-6}, то на данном шаге для слияния может быть выбрана любая из таких пар.

Вам предстоит промоделировать процесс слияния колоний и описать один из возможных вариантов развития событий.

Формат входного файла

Первая строка входного файла содержит целое число N — количество бактерий ($2 \le N \le 2014$).

Следующие N строк содержат описание матрицы расстояний $(d_{i,j})$. Каждая строка состоит из N символов. Символ $d_{i,j}$ в i-ой строке на j-ой позиции определяет расстояние между i-й и j-й бактериями.

Гарантируется, что $d_{i,i} = 0$, $d_{i,j} = d_{j,i}$, $0 \le d_{i,j} \le 9$ для всех i, $j = 1 \ldots N$.

Формат выходного файла

Занумеруем исходные колонии числами от 1 до N. Колония, которая получилась на i-ом шаге, $i = 1 \ldots N - 1$, будет иметь номер $N + i$.

В выходной файл необходимо вывести $N - 1$ строку. В i-ой строке должны содержаться три числа: номера объединившихся на i-м шаге колоний и расстояние между ними с абсолютной или относительной погрешностью не хуже 10^{-6} ($1 \le i \le N - 1$). Если вариантов ответа несколько, выведите любой из них.

Пример

input.txt	output.txt
4 0146 1025 4203 6530	1 2 1.000000 3 4 3.000000 5 6 4.250000

Задача 9. Треугольники (Division 1 Only!)

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

Петя учится на первом курсе университета. Особые проблемы ему доставляет аналитическая геометрия, ведь Петин ум просто сопротивляется всему принципиально новому.

Поэтому слушать лекции по этому предмету ему скучно. Тем не менее, эти лекции он зачем-то посещает, и как раз сейчас он находится на очередной. Он ищет, чем бы себя развлечь. Петя любит треугольники, потому что треугольники — это просто и привычно. У него с собой есть лист бумаги и синяя ручка. На листе бумаги он отметил n точек. Петя решил, что будет, соединяя эти точки линиями, рисовать и закрашивать синие треугольники.

И как раз в этот момент начался перерыв. Вернувшись после перерыва в аудиторию, он с досадой обнаружил, что кто-то нарисовал на его листе зелёный треугольник. Петя очень не любит зелёный цвет. Он хочет нарисовать свой синий треугольник с вершинами в отмеченных точках так, чтобы тот полностью накрыл зелёный треугольник. Тогда зелёный треугольник будет закрашен синим цветом, и Петя будет доволен. Зелёный треугольник должен лежать строго внутри синего, в частности, границы треугольников не могут иметь общих точек.

Поскольку лекции Петя не слушает, то сам решить эту задачу не может. Поэтому он обратился с ней к вам.

Формат входного файла

В первых трёх строках входного файла записаны через пробел по два целых числа — координаты вершин зелёного треугольника, заданные в порядке обхода против часовой стрелки. Гарантируется, что треугольник невырожденный.

В четвёртой строке задано целое число n — количество отмеченных точек ($1 \le n \le 10^5$). В каждой из следующих n строк содержится по два целых числа, разделённые пробелом — координаты сответствующей отмеченной точки. Гарантируется, что все n точек различны.

Все координаты во входном файле не превосходят 10^6 по абсолютному значению.

Формат выходного файла

Если искомого синего треугольника, в котором бы строго содержался зелёный треугольник, не существует, в выходной файл необходимо вывести слово "`NO`".

Иначе в первую строку нужно вывести слово "`YES`", а во вторую — номера точек из входного набора, которые являются вершинами искомого треугольника. Вершины должны быть перечислены в порядке обхода треугольника против часовой стрелки. Точки нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Если решений несколько, то выведите любое из них.

Примеры

input.txt	output.txt
0 0 1 0 0 1 3 -1 -1 2 0 -1 2	YES 1 2 3
0 0 1 0 0 1 3 0 -2 -2 1 3 1	NO

Задача 10. Заплыв

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

В заплыве за Скумбриевичем, который бултыхался около буйка, Великий комбинатор, как известно, плыл на боку, приглядывая за Берлагой, сидящим на берегу. Голова Остапа при этом всегда была повернута под одним и тем же углом. По этой причине плыл он не по прямой, а по кривой, так, что угол между направлением движения и направлением на Берлагу был все время постоянным. Это несколько удлиняло его траекторию.

Ваша задача — найти длину траектории движения Великого комбинатора.

Формат входного файла

Входной файл состоит из трех строк, каждая из которых содержит по два целых числа, не превосходящих по модулю 10^5. В первой строке записаны координаты точки первоначального положения Остапа Бендера, во второй — координаты положения Скумбриевича, цели его заплыва, а в третьей даны координаты Берлаги. Все три заданные точки различны. Гарантируется, что Берлага и Скумбриевич во время заплыва Остапа с места не двигались.

Формат выходного файла

В выходной файл необходимо вывести одно вещественное число — длину кратчайшей траектории от начального положения Остапа до буйка Скумбриевича, обладающей тем свойством, что касательная к этой траектории всегда сохраняет постоянный угол с направлением на Берлагу. Число выводить с абсолютной или относительной погрешностью не более 10^{-8}.

Пример

input.txt	output.txt
10 0 -10 0 0 0	31.4159265359

Комментарий

Для людей, не знакомых с творчеством Ильфа и Петрова: Остап Бендер и Великий комбинатор — это одно и то же лицо.

Задача 11. Faster Than Light

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 МиБ

В компьютерной игре “Faster Than Light” каждый корабль можно изобразить на плоском клеточном поле. Все клетки квадратные, сторона имеет единичную длину. Какие-то клетки поля являются отсеками корабля, по ним можно стрелять. Остальные клетки не принадлежат кораблю.

В игре есть лучевое оружие, которое стреляет следующим образом. При выстреле орудие проводит лучом по атакуемому кораблю вдоль некоторого отрезка фиксированной длины L.

Положение отрезка стреляющий может выбирать произвольно, при условии, что один из его концов находится внутри или на границе одного из отсеков атакуемого корабля. Другой конец отрезка может быть где угодно, в том числе за пределами клеточного поля.

Урон, нанесённый выстрелом кораблю и экипажу, зависит от набора отсеков, которые задеты лучом. Считается, что отсек задет лучом, если у отрезка и клетки отсека, включая ее границу, есть хотя бы одна общая точка.

Вам предлагается реализовать программу системы наведения, которая работает следующим образом. Для каждого отсека задано положительное число очков за попадание в него. Программа должна найти положение отрезка, которое даёт максимальную сумму очков по всем задетым отсекам.

Формат входного файла

В первой строке входного файла записано два целых числа N и M — количество строк и столбцов на поле соответственно($1 \le N, M \le 30$).

Во второй строке задано вещественное число L, запись которого содержит не более двух знаков после десятичной точки — длина отрезка ($0.1 \le L \le 50$).

Следующие N строк описывают клеточное поле. Каждая из них содержит по M целых чисел. Пусть в i-ой из этих строк j-ое число равно a_{ij} ($0 \le a_{ij} \le 10^7$, $i = 1, \ldots, N$, $j = 1, \ldots, M$). Если $a_{ij} = 0$, то соответствующая клетка пуста. Если $a_{ij} > 0$, то в клетке расположен отсек корабля, за попадание в который назначается a_{ij} очков.

Гарантируется, что на поле есть хотя бы одна клетка с отсеком корабля. Отсеки корабля могут быть не связаны друг с другом.

Формат выходного файла

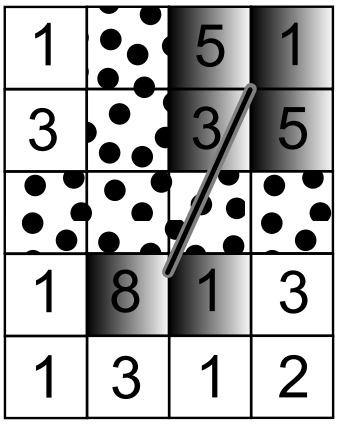
В выходной файл необходимо вывести одно целое число — максимально возможное количество очков за выстрел лучом.

Пример

input.txt	output.txt
5 4 2.5 1 0 5 1 3 0 3 5 0 0 0 0 1 8 1 3 1 3 1 2	23

Комментарий

В тесте из примера можно выстрелить так, чтобы задеть две клетки по 5 очков, клетки по 1 и 3 очков, а также клетки по 8 и 1 очков.



Задача 12. Проектирование Чанд Баори

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Индийский ступенчатый колодец Чанд Баори при некоторых допущениях представляет собой сужающуюся вниз пирамиду. Грани этой пирамиды состоят из рядов трапециевидных лестниц, по которым можно спуститься к воде. На первом уровне (в самом низу) находится одна лестница с двумя спусками слева и справа. На втором уровне находятся уже две такие лестницы и так далее.

Для посещения храма «Харшат Мата», построенного в честь богини радости и счастья, паломникам необходимо очиститься в водах колодца, то есть спуститься до самого низа. Паломники на своем пути к воде могут либо спускаться, либо проходить горизонтально вдоль какого-либо уровня, но не подниматься вверх.

Строители колодца хотят спроектировать его так, чтобы количество возможных путей спуска оказалось не меньше, чем число паломников. Если для двух путей существует ряд, в котором они отличаются лестницей, либо направлением спуска по одной и той же лестнице, то такие пути считаются различными.

Формат входного файла

В первой строке входного файла записано два целых числа N и M — количество уровней в колодце и число паломников, спускающихся по одной грани, соответственно ($1 \le N \le 20$, $0 \le M \le 1.5 \cdot 10^{18}$).

Формат выходного файла

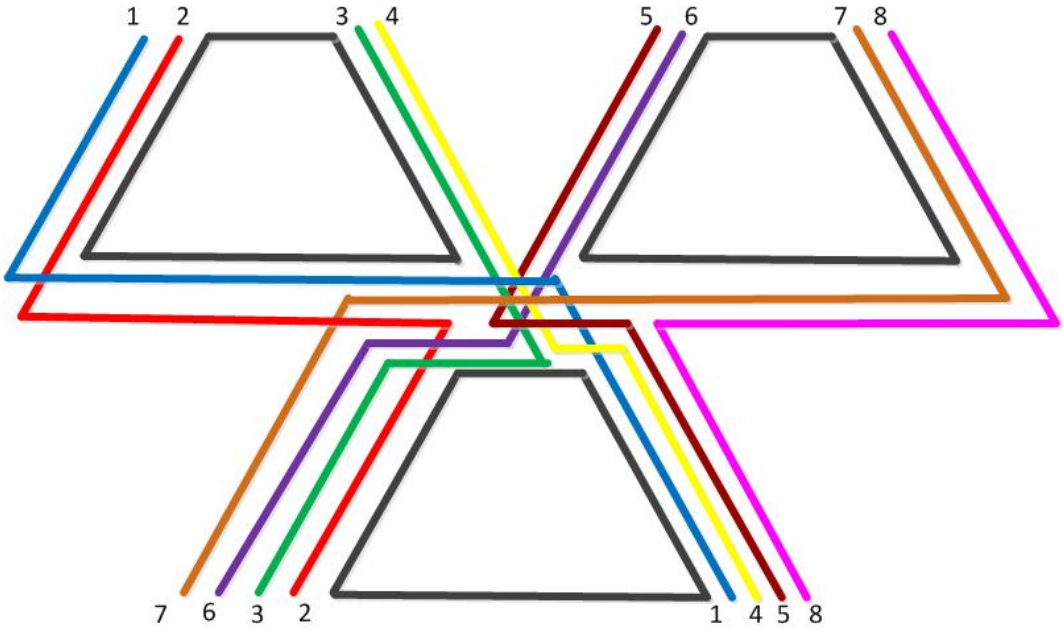
В выходной файл необходимо вывести “`Harshat Mata`”, если количество возможных путей для спуска по одной грани колодца оказалось не меньше числа паломников (также для одной грани), иначе вывести “`Nope`”.

Примеры

input.txt	output.txt
1 2	Harshat Mata
1 3	Nope
2 9	Nope
2 8	Harshat Mata

Комментарий

Для двухуровневого колодца существует восемь различных путей спуска, изображенных на рисунке (отрезки одного пути обозначены одинаковой цифрой).



Задача 13. Sum (Division 2 Only!)

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

Даны три целых числа A, K и P. Вычислите следующую сумму:

$$\sum_{i=1}^{K} A^i \ mod \ P$$

Формат входного файла

Первая и единственная строка входного файла содержит три целых числа A ($0 \le A \le 10^8$), K ($1 \le K \le 10^{16}$) и P ($1 \le P \le 10^8$), разделённые пробелами.

Формат выходного файла

Выведите одно число — значение требуемой суммы.

Примеры

input.txt	output.txt
3 4 101	120

Задача 14. Coinquerors (Division 2 Only!)

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

Правила старинной игры “Coinquerors” заключаются в следующем.

Каждый игрок участвует со своей монетой. Монета должна представлять собой окружность целого радиуса. Далее игроки бросают свои монеты так, что центр каждой монеты оказывается в точке с целыми координатами. После броска каждого игрока накрытый его монетой участок отмечается.

По завершении игры рассматриваются все отмеченные участки. Если у двух участников отмеченные участки пересекаются, то они объявляются союзниками. При этом гарантируется, что никакие два участка не имеют ровно одну общую точку (то есть случай касания соответствующих окружностей невозможен). Участник, набравший как можно больше союзников, и объявляется победителем.

Ваша задача — по координатам центров упавших монет и их радиусам определить победителя или же сказать, что игра завершилась вничью.

Формат входного файла

Первая строка входа содержит целое число T ($1 \le T \le 20$) — количество тестовых примеров.

Далее задаются тестовые примеры. Каждый тестовый пример начинается строкой, содержащей одно целое число N ($2 \le N \le 100$).

Каждая из последующих N строк содержит имя игрока, состоящее из не менее, чем двух и не более, чем из 255 строчных латинских букв, и три целых числа X, Y и R, задающих x и y координаты центра брошенной игроком монеты и её радиус ($-100 \le X, Y \le 100$, $1 \le R \le 10$).

Формат выходного файла

Для каждого тестового примера выведите одну строку с имеем победившего игрока. В случае, если победителей несколько, выведите строку “`TIE`”.

Пример

input.txt	output.txt
3 3 gennady 0 0 3 tomek 1 1 1 petr -1 -1 1 2 alice -100 -100 1 bob 100 100 100 3 john 0 0 10 john 2 2 3 jack -5 0 1	gennady TIE john