Задача A. Самая простая задача про кубики

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

Вчера Тая посещала музей. Экскурсия была долгой и интересной, но особенно ей понравилась комната, где были собраны коллекции кубиков 10 известных людей. Один из кубиков ей сильно приглянулся, но она забыла, кому он принадлежал. Зато она помнит как выглядели 3 видимые стороны кубика и кто какие кубики коллекционирует. По этой информации вам следует определить имена коллекционеров, в чьих коллекциях мог быть данный кубик.

У каждого кубика 6 граней. На каждой грани записано число от 1 до 6, на разных гранях разные числа. Числа могут изображаться точками, арабским числом или римским числом. Вдобавок каждая из граней покрашена в один из цветов — черный (Black), белый (White), зеленый (Green), желтый (Yellow), голубой (Skyblue), красный (Red), оранжевый (Orange) и фиолетовый (Purple).

Ниже представлен список имен коллекционеров и параметры кубиков, которым удовлетворяет вся его коллекция:

John	Все числа обозначены точками
David	Все числа обозначены не римскими числами
Peter	Кубик полностью белый
Robert	Грани кубика черные или белые
Mark	Нечетные числа на белом фоне, четные — на черном
Paul	Все простые числа арабские и все арабские числа простые
Patrick	Одноцветный цветной (не черный и не белый) кубик
Jack	Все римские числа на желтом фоне
Max	Все грани разноцветные
Alex	Числа одной формы записи на одном фоне, разного — на разном

Формат входных данных

Входной файл состоит их трех строк, описывающих видимые стороны кубика.

Первый символ i-й строки c_i ($c_i \in \{B, W, G, Y, S, R, O, P\}$) — цвет i-ой стороны (черный, белый, зеленый, желтый, голубой, красный, оранжевый и фиолетовый соответственно). Далее через пробел записана строка, обозначающая число на стороне в одном из трех форматов:

1. От 1 до 6 символов «`.`» (ASCII 46), означающих что число записано точками и равно количеству этих точек;
2. Арабское число от 1 до 6;
3. Римское число, записанное символами «`I`» (ASCII 73) и «`V`» (ASCII 86).

Гарантируется, что представленный кубик принадлежит хотя бы одному коллекционеру.

Формат выходных данных

Выходной файл должен содержать одну строку, содержащую имена коллекционеров, которым может принадлежать данный кубик. Имена должны быть записаны через пробел в любом порядке.

Все имена коллекционеров должны быть из следующего списка: `John`, `David`, `Peter`, `Robert`, `Mark`, `Paul`, `Patrick`, `Jack`, `Max`, `Alex`.

Примеры

input.txt	output.txt
W .. W ... W	John David Peter Robert Jack Alex
B 2 W 3 B 6	David Robert Mark Jack
G 1 G 2 G V	Patrick
G 2 G 3 Y	David Paul Jack Alex
W . B 2 W III	Robert Mark

Задача B. Идеальный подарок

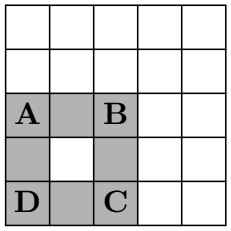
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Тая готовит подарок на день рождения. А самый лучший подарок — это подарок, сделанный своими руками. Недавно она научилось вышивать крестиком и решила воспользоваться этим умением для создания подарка.

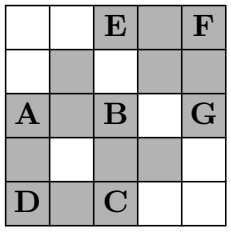
Дома у нее нашлась только канва, на которой ранее уже было вышито два крестика. Не беда — их можно дополнить до готовой картины. Опыта у нее мало, поэтому она выбрала довольно простой, но в то же время очень красивый рисунок, а именно параллелепипед. Ей хочется сделать подарок как можно скорее, поэтому количество дополнительных крестиков должно быть как можно **меньше**.

Параллелепипед на **бесконечном** клетчатом поле будем строить следующим образом.

Нарисуем прямоугольник $ABCD$ с верхним–левым углом A и нижним–правым C.



Проведем отрезки одинаковой длины вверх–вправо от вершин A, B и C — получим новые вершины E, F, G соответственно. Построим последние отрезки EF и FG.



Все длины сторон параллелепипеда должны быть **не менее 3-х клеток**.

Формат входных данных

В первой строке входного файла записано два целых числа x_1 и y_1 — координаты первого крестика. Во второй строке — координаты второго крестика: x_2, y_2. Координаты крестиков различны. Ось OX направлена слева направо, а ось OY направлена снизу вверх. Все числа принадлежат отрезку $[0, 10^9]$.

Формат выходных данных

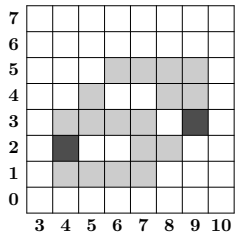
Выходной файл должен содержать одно число — **наименьшее** количество дополнительных крестиков.

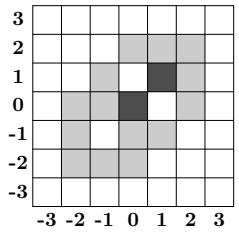
Примеры

input.txt	output.txt
4 2 9 3	17
0 0 1 1	14

Пояснение

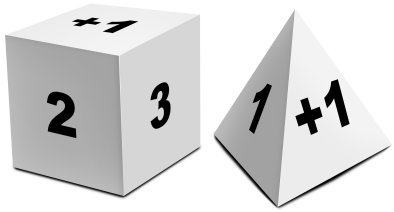
Примерам соответствуют следующие рисунки:





Задача C. Казино

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



Когда у Таи заканчиваются деньги, она идет играть в казино. Недавно в нем появилась новая игра, в которую Тая никак не может научиться играть оптимально. Помогите ей.

Игра проводится между крупье и посетителем казино. У крупье есть одна правильная k-гранная игральная кость, на гранях которой записаны все числа от 1 до k. В начале игры он берет кость и кидает один раз. Выпавшее число определяет количество очков крупье.

Для того чтобы выиграть, игроку требуется набрать больше очков, чем у крупье. Для этого ему предлагается выбор из n вариантов. Каждый вариант представляет собой пару: игральная кость и количество ее бросков. На каждой грани каждой кости написано какое-то число. Эта кость кидается нужное количество раз, все выпавшие числа суммируются и это значение и равно количеству очков игрока.

Но на некоторые грани, помимо обычных чисел, помечены как бонусные. Если выпала бонусная грань, то соотвествующее количество очков прибавляется к итогу, но за бросок это не считается. Все грани одной и той же кости попарно различны, то есть нет двух одинаковых бонусных граней и нет двух одинаковых обычных граней. На каждой кости есть по крайней мере одна грань, не являющаяся бонусной. Вероятность выпадения каждой грани одной игральной кости одинакова.

В задаче требуется для каждого возможного количества очков крупье от 1 до k определить номер варианта набора очков, при котором вероятность набрать строго больше очков, чем у крупье, максимальна.

Формат входных данных

Первая строка входного файла содержит одно целое число n ($2 \leqslant n \leqslant 10$) — количество игральных костей.

Следующие n строк содержат описания вариантов в следующем формате.

Первое число c_i ($1 \leqslant c_i \leqslant 10$) — количество бросков. Второе f_i ($2 \leqslant f_i \leqslant 12$) — количество граней. Следующие f_i строк v_{ij} — числа на гранях. v_{ij} является либо числом от 1 до f_i, обозначающим количество очков, либо числом от 1 до f_i с приписанным слева знаком плюс «+» (ASCII 43), обозначающим величину бонуса. В рамках одной кости все числа без знака плюс различны, все числа с приписанными плюсами различны, хотя бы одна грань не содержит знака плюс.

В последней строке записано одно целое число k, которое всегда равно $\max\limits_{1 \leqslant i \leqslant n} (c_i \times f_i)$.

Формат выходных данных

В выходном файле должно быть k строк, содержащих по одному целому числу b_i — номер наилучшего варианта, который позволит с наибольшей вероятностью набрать больше i очков (эта вероятность должна отличаться от правильного ответа не более чем 10^{-9}).

Кости нумеруются с 1 в том порядке, в котором перечислены во входном файле.

Примеры

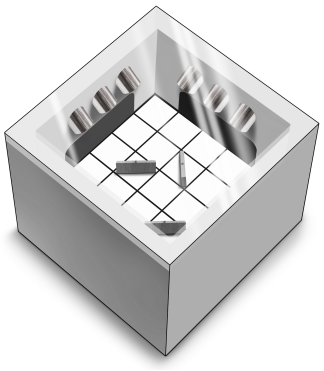
input.txt	output.txt
3 3 4 1 2 3 4 2 6 1 2 3 4 5 6 1 12 1 2 3 4 5 6 7 8 9 10 11 12 12	2 1 1 1 1 1 1 3 3 3 3 2
2 1 4 1 2 +1 +2 1 6 1 +1 2 3 4 5 6	2 2 2 2 1 1

Пояснение

В ответе первого примера на первом месте может быть число 1, а последнее может быть любым от 1 до 3.

Задача D. Головоломка

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

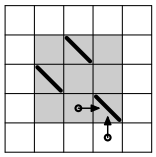


Тае как-то давно подарили головоломку, которую она никак не может решить.

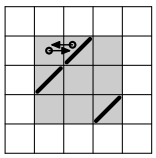
Она представляет собой поле $n \times n$, в каждой строке и в каждом столбце которой находится **ровно одна** перегородка, представляющая собой диагональный отрезок с началом в верхней левой точке и с концом в нижней правой точке. После включения головоломки в некоторые целые моменты времени из трубок, которые расположены по **краям сетки**, вылетают шарики. За один момент времени шарик перемещается в соседнюю по стороне клетку. Шарик, врезаясь в перегородку, меняет свое направление на 90°. Шарик исчезает после пересечения им границы поля. Чтобы решить головоломку, необходимо до её включения повернуть некоторые перегородки на 90° относительно их центра, чтобы после этого никакие два шарика не столкнулись внутри поля.

Два шарика сталкиваются если:

1. Они находятся в один и тот же целый момент времени в одной и той же клетке поля (если в этой клетке расположена перегородка, то для столкновения шарики должны находиться по одну сторону от нее).



2. Они встретились на границе сетки (граница поля тоже считается местом столкновения).



В этой задаче требуется найти **любое** решение этой головоломки.

Формат входных данных

Первая строка входного файла содержит одно целое число n ($1 \leqslant n \leqslant 500$) — размер поля.

Следующая строка содержат n целых чисел ($1 \leqslant c_i \leqslant n$) — номер колонки i-й перегородки, у которой номер строки равен i. Все номера колонок различны.

В третьей строке написано одно целое число m ($1 \leqslant m \leqslant 10^4$) — количество шаров.

Следующие m строк содержат по 3 целых числа x_i, y_i, t_i ($0 \leqslant t_i \leqslant 10^8$), описывающих моменты вылета шаров — в момент времени t_i шар появится в клетке (x_i, y_i), которая прилегает к границе поля. Моменты идут в порядке неубывания t_i. Координаты (x_i, y_i) могут быть в одной из четырех областей:

1. $x_i = 0$, $1 \leqslant y_i \leqslant n$;
2. $1 \leqslant x_i \leqslant n$, $y_i = 0$;
3. $x_i = n + 1$, $1 \leqslant y_i \leqslant n$;

4. $1 \leqslant x_i \leqslant n$, $y_i = n + 1$.

Гарантируется, что при данных значениях ответ существует.

Формат выходных данных

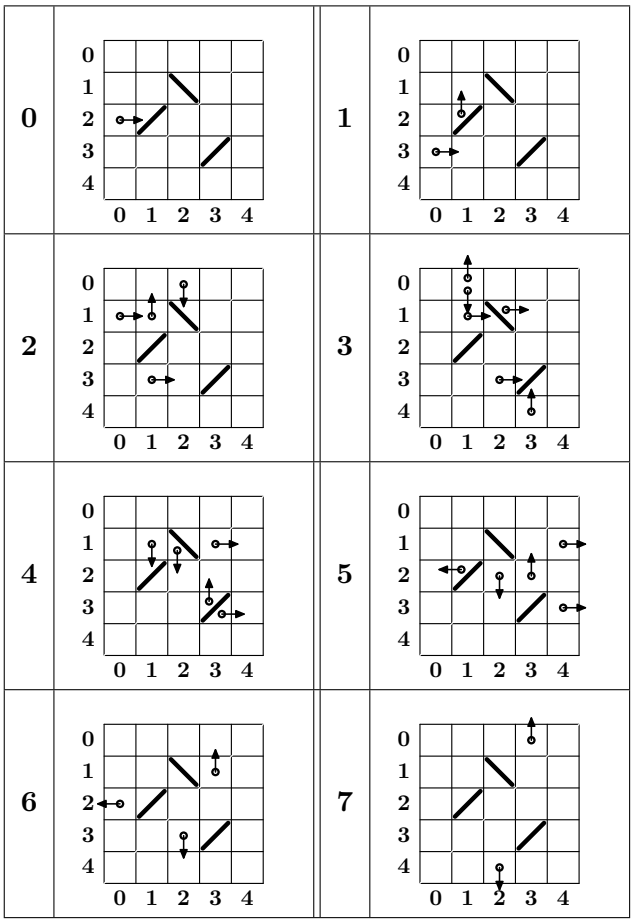
Выходной файл должен содержать одну строку из 0, 1. i-й символ равен 0, если i-ую перегородку поворачивать не надо, 1 — если надо.

Примеры

input.txt	output.txt
3 2 1 3 6 2 0 0 3 0 1 1 0 2 0 2 2 4 3 3 0 1 3	011

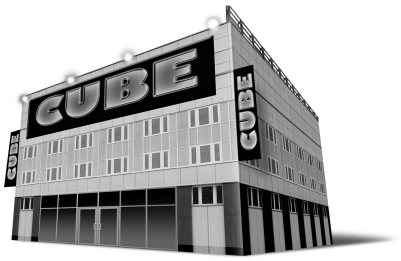
Пояснение

Ниже показаны положения шариков для примера в зависимости от времени.



Задача E. В кубе

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



Тая любит ходить с друзьями в кафе «В кубе», ведь там очень удобная система обслуживания. Чтобы сделать заказ, посетитель должен подойти к специальному стенду и выбрать понравившиеся блюда. Несколько таких стендов расположены на территории кафе в фиксированном положении.

В кафе посетители сидят за столами, которых k. За i-м столом может сидеть c_i человек. Неудобностью размещения стола назовем сумму расстояний от него до c_i ближайших стендов.

Формально, кафе представляет собой сетку $(0, 0) - (5000, 5000)$. В каждой клетке (x, y) $(0 \leqslant x, y \leqslant 5\,000)$ может находиться один стенд, один стол, либо ничего.

Расстояние между клетками (x_1, y_1) и (x_2, y_2) равно $|x_2 - x_1| + |y_2 - y_1|$.

Требуется расставить столы так, чтобы сумма неудобностей всех столов была минимальной.

Формат входных данных

Первая строка входного файла содержит два целых числа n и k $(1 \leqslant n \leqslant 18,\ 1 \leqslant k \leqslant 200)$ — количество стендов и столов соответственно.

Следующие n строк содержат координаты i-го стенда: два целых числа x_i и y_i $(0 \leqslant x_i, y_i \leqslant 5\,000)$.

Ось OX направлена слева направо, а ось OY направлена снизу вверх.

Далее k строк содержат по одному целому числу c_j $(1 \leqslant c_j \leqslant n)$ — количеству мест j-го стола.

Формат выходных данных

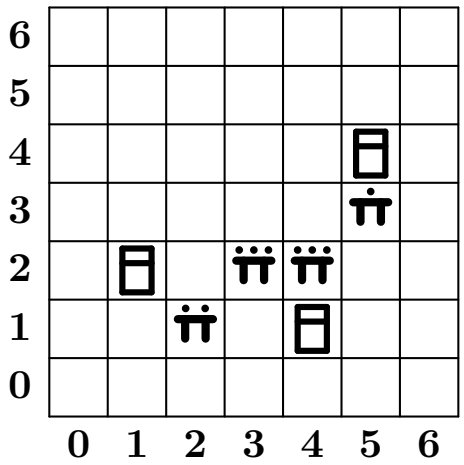
Выходной файл должен содержать одно целое число — минимальную сумму неудобностей.

Примеры

input.txt	output.txt
3 4 1 2 4 1 5 4 1 2 3 3	20
2 10 0 0 5000 5000 1 1 1 1 1 1 1 1 1 1	16

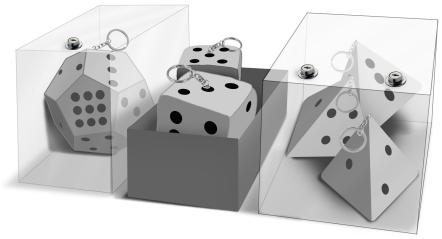
Пояснение

Возможный вариант расположения столов для первого примера выглядит так:



Задача F. Магазин игрушек

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



Тая часто проходила мимо магазина игрушек и видела висящее около витрины электронное табло с двумя целыми числами. На витрине магазина лежат несколько видов игрушек, но числа на табло могут не соответствовать количеству видов игрушек. Оказалось, что не все игрушки с витрины можно купить, потому что их убирают с витрины не сразу, а спустя некоторое время после покупки. Для разных видов игрушек это время может быть разное.

Пусть есть n видов игрушек. Для каждого вида игрушки известно первоначальное количество c_i и количество минут t_i после покупки, через которое игрушка этого вида убирается с витрины. Каждую минуту происходит следующее:

- с витрины убирают игрушки, купленные соответствующее время назад;
- на табло обновляются числа;
- приходит новый покупатель и **обязательно покупает какую-то игрушку, оставшуюся в наличии**.

Таю всегда интересовал смысл чисел на табло и совсем недавно она его узнала. Оба числа показывали, сколько видов игрушек можно было приобрести в магазине, но первое показывало количество видов игрушек, **возможно** не распроданных к данному моменту времени, а второе — количество видов игрушек, **гарантированно** не распроданных к данному моменту времени. Также Тае интересно, насколько это табло информативно для посетителей. Поэтому ей нужна программа, которая будет моделировать поведение покупателей и пересчитывать табло.

Ваша задача — для каждой минуты определить числа на табло.

Формат входных данных

Первая строка входного файла содержит одно целое число n ($1 \leqslant n \leqslant 10^5$) — количество видов игрушек.

Следующие n строк содержат по два целых числа c_i и t_i ($1 \leqslant c_i \leqslant 10^5$, $1 \leqslant t_i \leqslant 100$) — количество игрушек i-го вида и время, через которое игрушку убирают с витрины после покупки соответственно.

Далее следует строка из одного целого числа k ($1 \leqslant k \leqslant 10^5$) — количества покупателей.

Следующие k строк содержат по одному целому числу q_i и q_i целых чисел $p_1, p_2, ..., p_{q_i}$ — количество игрушек, убранных в i-ую минуту и номера этих игрушек.

Формат выходных данных

Выходной файл должен содержать k строк, состоящих из двух целых чисел a_i и b_i — первое и второе число на табло в момент начала i-ой минуты соответственно.

Примеры

input.txt	output.txt
3 1 2 2 1 3 3 6 0 0 0 3 1 2 3 0 1 2	3 3 3 2 3 2 2 2 2 2 1 1

Пояснение

В приведенном выше примере на витрине находятся одна игрушка первого вида, две игрушки второго вида и три игрушки третьего вида, которые убирают после покупки через 2, 1 и 3 минуты соответственно. Числа на табло должны меняться в следующем порядке:

- 3/3: перед первым покупателем никого не было, он может купить любую игрушку.
- 3/2: первый покупатель мог купить игрушку первого вида, поэтому нет уверенности, что второй клиент может ее купить.
- 3/2: так как ни игрушку первого вида не убрали с витрины, ни второго, то это означает, что первый покупатель купил игрушку третьего вида. Что купил второй — пока определить нельзя.
- 2/2: игрушек первого вида больше не осталось.
- 2/2: никакую игрушку с витрины не убрали, значит предыдущий покупатель купил игрушку третьего вида.
- 1/1: Некупленной осталась только одна игрушка третьего вида.

Задача G. Доставка подарков

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

У Таи на работе случилась неприятность: заболел водитель, а надо срочно отвезти подарки из одного магазина в другой. Но, к счастью, сейчас у нее перерыв, и этот магазин находится на этой же улице, поэтому ее умения ехать только вперед с постоянной скоростью v_1 вполне достаточно, чтобы помочь в сложившейся ситуации.

Однако на одном из перекрестков по пути к магазину перестал работать светофор, и сейчас там стоит регулировщик, которому нельзя уходить с места.

В некий момент времени он заметил движущийся на него грузовик, который совсем не собирался сворачивать. Двинешься с места — получишь штраф, но двигаться придется. Поэтому регулировщик хочет пропустить грузовик так, чтобы **минимизировать время, которое он не будет находится в первоначальном положении**. Регулировщик может двигаться как угодно, но его скорость не должна превышать v_2.

Представим грузовик в виде прямоугольника, а регулировщика в виде точки. Требуется, чтобы ни в какой момент времени точка не оказалась строго внутри прямоугольника, и время, которое точка не будет находится в (p, q) (координаты первоначального положения) было минимально возможным.

Формат входных данных

Первая строка содержит 6 целых чисел a, b, p, q, v_1, v_2 ($1 \leqslant a \leqslant 100$, $0 \leqslant b \leqslant 99$, $-a < p < a$, $b < q \leqslant 100$, $1 \leqslant v_1, v_2 \leqslant 100$). В начальный момент времени верхний левый угол грузовика имеет координаты $(-a, b)$, нижний правый — $(a, 0)$. Регулировщик в начальный момент времени находится в точке (p, q). Грузовик движется в сторону увеличения второй координаты с постоянной скоростью v_1. Максимальная скорость регулировщика — v_2. При $b = 0$ считать длину грузовика сколь угодно малой.

Все расстояния измеряются в метрах, скорости даны в метрах в секунду.

Гарантируется, что значения буду таковы, что ответ не превысит 10 000.

Формат выходных данных

Выходной файл должен содержать одно вещественное число — наименьшее возможное время, когда регулировщик будет отсутствовать в точке (p, q). Ответ должен отличаться от правильного не более чем на 10^{-6} по абсолютной или относительной погрешности.

Примеры

input.txt	output.txt
4 0 1 5 1 1	6
3 2 -1 10 5 2	2.306019375

Пояснение

В первом примере оптимальным действием будет подождать 2 секунды, затем двигаться 3 секунды вправо на максимальной скорости, и после этого двигаться назад влево на максимальной скорости.

Задача H. Игра с кубиками

Имя входного файла:	`Стандартный поток ввода`
Имя выходного файла:	`Стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт
Ограничение на количество запросов:	7 500

Это интерактивная задача.

Тая нашла на чердаке очень старую игру, в которую она выигрывала раз через раз. Покажите ей, как можно выигрывать в нее гарантированно.

Набор игры состоит из круглой фишки радиуса 1, у которой сверху нарисована стрелка, двух кубиков и набора наклеек. Всего в наборе 360 наклеек со всеми целыми значениями от 0° до 359°.

Перед началом игры необходимо на столе пометить некую точку, положить фишку на стол, выбрать 12 различных наклеек, 6 их них наклеить на первый кубик, а остальные шесть — на второй. Цель игры — накрыть точку фишкой. Этого нужно достичь, делая ходы по следующему правилу. В начале хода выбирается один из двух кубиков, кидается, фишка поворачивается на выпавшее число градусов против часовой стрелки и затем перекладывается по направлению стрелки на расстояние 10.

Помеченная точка всегда имеет координаты $(0, 0)$. Начальное положение центра фишки (x, y) удовлетворяет ограничению

$$2 \leqslant \max(|x|, |y|) \leqslant 500$$

Количество запросов для этой задачи равно количеству сделанных ходов.

Протокол взаимодействия

В начале программа жюри выводит координаты центра фишки и ее вектор скорости. Далее программа участника должна сказать, какие числа наклеены на два кубика. Затем на каждый вывод программой участника номера кубика, программа жюри выводит выпавшее значение градусов и сообщает, достигла ли фишка требуемой цели. Если да, то программа участника должна завершиться, иначе программа жюри выводит новое расположение точки и новый вектор скорости и процесс повторяется.

Формат выходных данных

Первые две строки выходного потока должны содержать по 6 целых чисел от 0 до 359 — наклейки для первого и второго кубика соответственно. Все числа в этих строках должны быть различными.

Следующие строки должны содержать по одному целому числу 1 или 2 — номер кубика, который необходимо бросить.

Не забывайте использовать команду `flush` после вывода каждой строки.

Формат входных данных

Входной поток состоит из четверок строк:

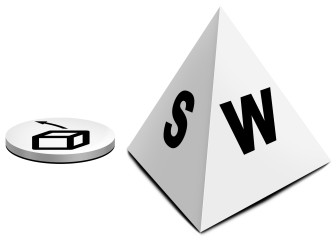
1. x, y — координаты центра фишки;
2. v_x, v_y ($v_x^2 + v_y^2 = 10$) — направление стрелки фишки;
3. d — выпавшее число градусов (веронятность выпадения каждого числа из накленных на соответствующий кубик одинакова);
4. «`Yes`» — после последнего движения фишка накрыла точку $(0, 0)$, «`No`» — не накрыла.

Примеры

Стандартный поток вывода	Стандартный поток ввода
	10.000000000 -10.000000000 0.000000000 -10.000000000
180 96 250 187 319 6 295 152 82 90 32 334 1	
	180 No 10.000000000 0.000000000 0.000000000 10.000000000
2	
	90 Yes

Задача I. Игра с монетками

Имя входного файла:	`Стандартный поток ввода`
Имя выходного файла:	`Стандартный поток вывода`
Ограничение по времени:	2 секунды (3 для Java)
Ограничение по памяти:	256 мегабайт
Ограничение на количество запросов:	60 000



Это интерактивная задача.

Тая может с легкостью победить в этой игре, а все те, кому она предлагала в нее сыграть, пока ни разу не смогли дойти до финиша. Теперь она предлагает вам сыграть в эту игру.

В набор игры входит сетчатое поле $n \times n$ ($5 \leqslant n \leqslant 40$), фишка, две монетки (`COIN1`, `COIN2`), две игральные кости (`DICE1`, `DICE2`), на которых написаны стороны света, и большое количество блоков размером в одну клетку.

Название	Количество граней	Значения на гранях
`COIN1`. Монетка движения	2	SLIDE, RAM
`COIN2`. Монетка изменения лабиринта	2	PLACE, REMOVE
`DICE1`. Пирамидка с направлениями	4	`N` (Север), `S` (Юг), `W` (Запад), `E` (Восток)
`DICE2`. Октаэдр с направлениями	8	`N` (Север), `S` (Юг), `W` (Запад), `E` (Восток), `NW` (Северо–запад), `NE` (Северо–восток), `SW` (Юго–запад), `SE` (Юго–восток)

Перед началом игры в некоторые клетки сетки помещаются фишка и блоки. Затем делаются ходы такого вида: сначала игрок выбирает одну из монет и кидает ее для определения действия, затем выбирает одну из костей и кидает ее для получения направления *dir*. Далее необходимо сделать одно из четырех действий в зависимости от того, что выпало на монетке.

Название	Действие
`SLIDE`	Двигать фишку по пустым клеткам в направлении *dir*, пока фишка не упрется в блок или в край сетки
`RAM`	Двигать фишку по пустым клеткам в направлении *dir* до первого попавшегося блока, затем двигать фишку вместе с этим блоком в этом же направлении, пока движимый блок не упрется в другой блок или в край сетки
`PLACE`	Если соседняя клетка в направлению *dir* от фишки не за пределами сетки и пуста, поместить туда блок
`REMOVE`	Если соседняя клетка в направлению *dir* от фишки не за пределами сетки и содержит блок, убрать этот блок

Цель — переместить фишку в конечную клетку.

Запросом в этой задаче является один ход — бросок одной монетки и одной кости с направлением.

Протокол взаимодействия

В начале программа жюри выводит размер лабиринта, сам лабиринт и координаты конечной клетки. Далее следуют действия, состоящие из следующих этапов:

1. Программа жюри выводит координаты фишки или сообщает, что фишка достигла конечной клетки.

2. Программа участника выводит номер кидаемой монеты.
3. Программа жюри говорит, что выпало на ней.
4. Программа участника выводит номер кидаемой кости.
5. Программа жюри выводит выпавшее направление и дополнительную информацию для действия «RAM».

Формат выходных данных

Выходной поток должен состоять из пар строк «название монетки — название кости». Название монеты — это строка COIN1 или COIN2. Название кости — DICE1 или DICE2.

Не забывайте использовать команду flush после вывода каждой строки.

Формат входных данных

Первая строка входного потока содержит одно число n — размер лабиринта. Следующие n строк содержат по n символов «.» (ASCII 46) или «#» (ASCII 35), обозначающих свободную и занятую клетку соответственно.

Следующая строка содержит два целых числа r_f и c_f — координаты конечной клетки, номер строки и номер столбца соответственно. Верхий–левый угол имеет координаты $(1, 1)$. Нижний–правый — (n, n). В начальной и конечной клетке блоков нет.

Далее идут группы строк, описывающие каждый ход.

- Первая строка группы содержит два целых числа r, c — номер строки и столбца, где находится фишка, или $(-1, -1)$, если фишка находится в конечной клетке.
- В следующей строке находится название выпавшей команды — SLIDE, RAM, PLACE или REMOVE.
- Далее идет строка с выпавшим направлением — N, S, W, E, NE, NW, SW или SE.
- Если текущее действие «RAM», то следующие две строки содержат по два целых числа r_1, c_1 и r_2, c_2, означающих, что фишка стукнулась о блок с координатами (r_1, c_1), а затем перетащила его на клетку (r_2, c_2).

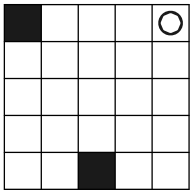
Движение на север соответствует уменьшению номера строки, на юг — увеличению номера строки, запад — уменьшению номера столбца, восток — увеличению номера столбца.

Вероятность выпадения каждой команды и каждого направления для одной игральной кости одинакова.

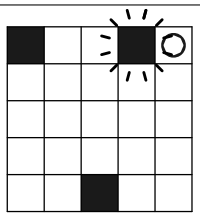
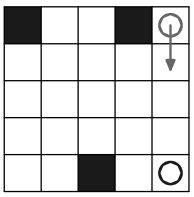
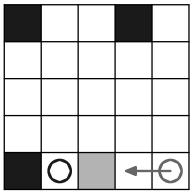
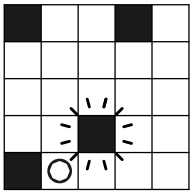
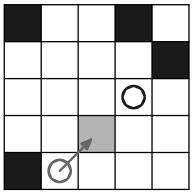
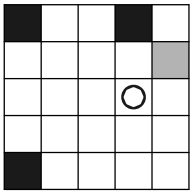
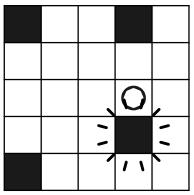
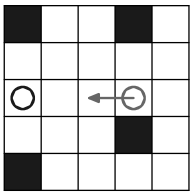
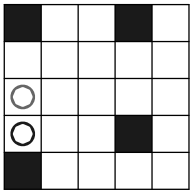
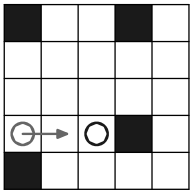
Примеры

Стандартный поток вывода	Стандартный поток ввода
	5 #.... ....#.. 4 3 1 5
COIN2 DICE1	PLACE W 1 5
COIN1 DICE1	RAM S 6 5 6 5 5 5
COIN1 DICE1	RAM W 5 3 5 1 5 2
COIN2 DICE2	PLACE NE 5 2
COIN1 DICE2	RAM NE 4 3 2 5 3 4
COIN2 DICE2	REMOVE NE 3 4
COIN2 DICE1	PLACE S 3 4
COIN1 DICE1	SLIDE W 3 1
COIN1 DICE1	RAM S 5 1 5 1 4 1
COIN1 DICE1	SLIDE E -1 -1

В примере первоначальное состояние лабиринта выгдядит следующим образом:

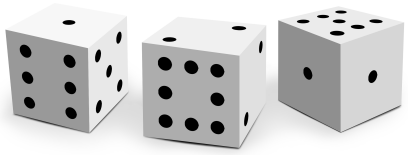


Получившийся в примере путь изображен ниже:

COIN2 → PLACE DICE1 → W		COIN1 → RAM DICE1 → S	
COIN1 → RAM DICE1 → W		COIN2 → PLACE DICE2 → NE	
COIN1 → RAM DICE2 → NE		COIN2 → REMOVE DICE2 → NE	
COIN2 → PLACE DICE1 → S		COIN1 → SLIDE DICE1 → W	
COIN1 → RAM DICE1 → S		COIN1 → SLIDE DICE1 → E	

Задача J. Самая сложная задача про кубики

Имя входного файла:	`Стандартный поток ввода`
Имя выходного файла:	`Стандартный поток вывода`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт



Это интерактивная задача.

Тая очень хорошо играет в игру, которую сама придумала. У вас есть уникальная возможность поиграть с ней и попытаться выиграть.

Игра включает в себя два одинаковых набора из n ($2 \leqslant n \leqslant 10$) 6-гранных кубиков, на каждой грани которых записано число от 1 до 100. Игроки играют одновременно, независимо и не знают о состоянии игры друг друга.

Игра происходит следующим образом. Игрок выбирает любой кубик из набора и кидает его. Выпадает число. Он может принять его (это будет количеством очков игрока) или кинуть другой кубик, но тогда начислится 1 очко штрафа. Один и тот же кубик кидать нельзя в течение одной игры. Итоговое количество очков игрока равно разнице последнего числа и количества повторных бросков. Игра заканчивается, когда оба игрока решают принять выпавшее число.

Так как Тая уже несколько лет играет в эту игру, она будет немного поддаваться. Вы будете считаться победителем в игре, если ваше количество очков **больше или равно** количеству очков у нее. Также Тая будет придерживаться одной и той же стратегии в рамках одного теста: она будет кидать кубики всегда в одном и том же порядке. А принимать решение о перебрасывании так: если, продолжая кидать кубики в заданом порядке, она с вероятностью не менее 50% может набрать очков больше, чем выпало на последнем кубике, с учетом штрафа, то она продолжает играть, иначе останавливается.

В рамках этой задачи вы должны сыграть с Таей 10 000 игр, причем количество выигранных вами игр должно быть не менее 5 000.

Протокол взаимодействия

Сначала программа жюри выводит описание кубиков. Далее программа участника должна сыграть с программой жюри 10 000 игр. Каждая игра проходит по следующему сценарию. Программа участника выводит номер брошенного кубика. На это программа жюри выдает количество набранных очков с учетом штрафа. В ответ программа участника сообщает, принято ли выпавшее число. После завершения игры, программа жюри сообщает результат игры — выиграла программа участника или нет. Затем начинается новая игра.

Формат выходных данных

Для броска кубика выходной поток должен содержать строку, состоящую из одного числа от 1 до n — номера кубика. После каждого броска поток должен содержать строку «`Yes`», если программа соглашается с набранными очками, иначе «`No`».

Не забывайте использовать команду `flush` после вывода каждой строки.

Формат входных данных

Первая строка входного файла содержит одно целое число n — количество кубиков.

Следующие n строк содержат по 6 целых чисел от 1 до 100 — числа на гранях i-го кубика.

На каждый бросок кубика входной поток содержит одно целое число — выпавшее число. Вероятность выпадения каждой грани кубика одинакова.

После завершения очередного раунда входной поток содержит одну строку — «`Win`», если программа участника выиграла или «`Lose`», если проиграла.

Примеры

Стандартный поток вывода	Стандартный поток ввода
1 No 2 No 3 Yes 2 Yes	3 1 2 3 4 5 6 2 2 2 8 8 8 1 1 1 7 7 7 1 1 5 Lose 8 Win

Пояснение

В примере показана серия только из 2 игр. При реальном тестировании будут сыграны все положенные 10 000 игр.

В этом тесте Тая кидает кубики в том порядке, в котором они перечислены во входном потоке.