Задача A. Почти палиндромы

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

Будем называть число x почти палиндромом, если его десятичная запись может быть превращена в палиндром изменением не более чем одной цифры. Например, числа 1234321, 1234311 и 123421 являются почти палиндромами, а 1234213 или 12345331 — нет.

По заданному числу a найдите количество чисел x, таких что $1 \le x \le a$, и x является почти палиндромом.

Формат входного файла

Входной файл содержит несколько тестовых примеров. В каждом тестовом примере задано число a ($1 \le a \le 10^{18}$). Последняя строка входного файла содержит число 0, ее не надо обрабатывать.

Формат выходного файла

Для каждого числа a во входном файле выведите в выходной файл одну строку, содержащую количество почти палиндромов, не превосходящих a.

Пример

almost.in	almost.out
10 1000 1000000 0	10 1000 45010

Задача B. Декартово дерево

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

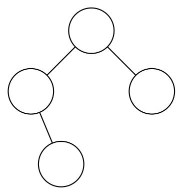
Рассмотрим один специальный тип двоичных деревьев поиска, который часто называют "декартовым деревом". Напомним, что двоичным деревом поиска называется двоичное дерево, в каждой вершине которого записан некоторый ключ (в этой задаче ключ представляет собой целое число), причем для каждой вершины u выполнены следующие условия: все ключи в левом поддереве u меньше чем ключ в вершине u, а все ключи в правом поддереве — больше.

Будем называть двоичное дерево поиска декартовым деревом, если в каждой вершине u помимо основного ключа x_u хранится также вспомогательный ключ y_u, причем для этих ключей выполнено "условие кучи": если v — родитель u, то $y_v < y_u$.

Будем называть множество пар $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ *корректным*, если все x_i — различные числа от 1 до n, и то же условие выполнено для y_i. Легко показать, что для любого корректного множества пар существует в точности одно декартово дерево, которое содержит пары из заданного множества в качестве пар ключей.

Рассмотрим двоичное дерево T с n вершинами. Ваша задача — найти количество различных корректных множеств пар, таких что их можно расставить в вершинах T, чтобы превратить его в декартово дерево.

Например, для дерева, приведенного на рисунке, существует три соответствующих корректных множества пар: $\{(1,2),(2,3),(3,1),(4,4)\}$, $\{(1,2),(2,4),(3,1),(4,3)\}$ и $\{(1,3),(2,4),(3,1),(4,2)\}$.



Формат входного файла

Первая строка входного файла содержит n — количество вершин в дереве T ($1 \le n \le 200$). Следующие n строк описывают его вершины. Каждая вершина описывается двумя числами: номерами левого и правого ребенка. Если один из детей отсутствует, то вместо его номера указан 0. Гарантируется, что описание дерева корректно. Корень дерева имеет номер 1.

Формат выходного файла

Выведите одно число — количество различных корректных множеств пар, которые можно расставить в вершинах T, чтобы получилось декартово дерево.

Пример

cartesian.in	cartesian.out
4 2 3 0 4 0 0 0 0	3

Задача C. Анализ ДНК

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

Биологи Карельского Мутационного Проекта (КМП) недавно решили начать новые исследования, которые должны доказать, что люди — близкие родственники мамонтов. Чтобы доказать это странное предположение, ученые планируют сравнить ДНК людей и мамонтов.

Для сравнения ДНК разделяется на фрагменты длины n и они последовательно сравниваются. Поскольку в процессе развития у людей и мамонтов могли происходить мутации, предлагается следующий способ сравнения фрагментов.

Рассмотрим строку α. Будем говорить, что α *мутирует* в β, если $\alpha = xyz$ для некоторых (возможно пустых) x, y и z, а $\beta = xy^Rz$, где y^R означает строку y, записанную в задом наперед (например, “abc”^{R} =“cba”). Будем говорить, что строки α и β *похожи*, если α может быть превращена в β не более чем за 4 мутации.

По двум данным фрагментам ДНК определите, похожи ли они.

Формат входного файла

Входной файл содержит две строки, состоящие из символов ‘A’, ‘D’, ‘G’ и ‘T’. Строки имеют одинаковую длину, не превышающую 30.

Формат выходного файла

Выведите в выходной файл “Similar”, если строки похожи, и “Different”, если нет.

Пример

dna.in	dna.out
ATGAATGA AGGAATTA	Similar
ATGAATGAATGA TTTAAAAAAGGG	Different

В первом примере возможна следующая последовательность мутаций: “ATGAATGA”→“AGTAAGTA”→“AGGAATTA”.

Задача D. Jивой Jурнал

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

Программист Саша участвует в создание блог-сервиса Jивой Jурнал. Планируется, что этот сервис будет предоставлять гораздо больше возможностей, чем известный всем LiveJournal. В настоящее же время проблему составляет реализация всех базовых возможностей LiveJournal'a. Одной из таких возможностей является поддержка списков друзей для пользователей.

Заданы: список пользователей, являющихся друзьями данного пользователя, и список пользователей, у которых данный пользователь содержится в списке друзей.

Необходимо получить список друзей данного пользователя (`Friends`), список его взаимных друзей (`Mutual Friends`), и список тех пользователей, у кого данный пользователь содержится в списке друзей, но которые не являются его взаимными друзьями (`Also Friend of`).

Формат входного файла

Первая строка входного файла содержит число n $(0 \le n \le 200)$ друзей данного пользователя. Последующие n строк содержат каждая по одному имени пользователя, который является другом данного. $(n+2)$-ая строка содержит число m $(0 \le m \le 200)$ пользователей, у которых данный содержится в списке друзей. Далее заданы имена пользователей, у которых данный находится в списке друзей. Эти пользователи заданы в том же формате, что и друзья данного.

Имена пользователей — строки длиной не более 20 символов, содержащие только строчные буквы латинского алфавита и символы тире ("-"). Каждый пользователь указан не более одного раза в каждом из списков.

Формат выходного файла

В выходной файл список друзей данного пользователя (Friends), список его взаимных друзей (Mutual Friends), и список тех пользователей, у кого данный пользователь содержится в списке друзей, но которые не являются его взаимными друзьями (Also Friend of). В каждом списке пользователи должны быть отсортированы по алфавиту. Следуйте формату, приведенному в примерах.

Примеры

jj.in
```
3
vasya-pupkin
bill-hates
ivan-ivanov
2
vasya-pupkin
destroyer
```

jj.out
```
Friends: bill-hates, ivan-ivanov, vasya-pupkin
Mutual Friends: vasya-pupkin
Also Friend of: destroyer
```

jj.in
```
0
0
```

jj.out
```
Friends:
Mutual Friends:
Also Friend of:
```

Задача E. Фонарь

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

«Одна голова хорошо, а две лучше. Одна лампочка хорошо, а две лучше,» — подумал Миша, и решил собрать фонарик с двумя лампочками. Теперь он хочет узнать, насколько фонарик с двумя лампочками лучше, чем фонарик с одной. Для этого Миша посветил фонариком на стену, и каждая из лампочек осветила на ней круг.

Эффективность фонарика Миша хочет оценить через площадь освещенной части стены. Миша догадался измерить координаты центров освещенных кругов и их радиусы (которые оказались одинаковыми), но что делать дальше он не знает. Напишите программу, которая поможет Мише.

Формат входного файла

Во входном файле записаны целые числа x_1, y_1, x_2, y_2 и r ($1 \le x_1, y_1, x_2, y_2, r \le 100$) — координаты центров кругов и их радиус соответственно.

Формат выходного файла

В выходной файл выведите площадь, покрытую двумя кругами с точностью не менее трех знаков после десятичной точки.

Примеры

light.in	light.out
1 2 3 4 2	22.84955592153876

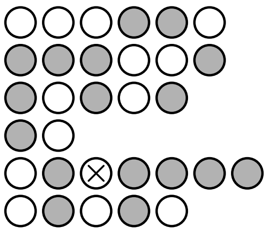
Задача F. Игра с камнями

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

Андрюша и Миша играют в увлекательную игру с разноцветными камнями. Сначала они выкладывают несколько рядов красных и синих камней на столе. Андрюша играет синими камнями, а Миша — красными.

Игроки делают ходы по очереди. Андрюша ходит первым. Своим ходом игрок выбирает один из камней своего цвета и убирает со стола этот камень, а также все камни, лежащие в том же ряду справа от него. Отметим, что хотя выбранный камень должен быть цвета того игрока, который делает ход, камни справа от него убираются со стола вне зависимости от их цвета. Если камней своего цвета на столе нет, то игрок, который должен делать ход, проигрывает.

Рассмотрим, например, начальную позицию, показанную на рисунке. Синие камни изображены как незакрашенные кружки, а красные — как закрашенные.



В этой позиции Андрюша может выиграть. Своим первым ходом он берет третий камень в пятом ряду (на рисунке он помечен крестиком). После этого он использует симметричную стратегию: если Миша делает ход в первом ряду, Андрюша повторяет его ход во втором, и наоборот; если Миша делает ход в третьем ряду, то Андрюша повторяет его ход в шестом ряду, и т. д.

По начальной позиции в игре, определите кто выигрывает при оптимальной игре обоих игроков, и если это Андрюша, то какой должен быть его первый ход.

Формат входного файла

Первая строка входного файла содержит n — количество рядов камней ($1 \le n \le 100$). Следующие n строк описывают выложенные ряды. Каждая строка содержит последовательность букв '`B`' и '`R`', которые обозначают синие и красные камни, соответственно. Каждый ряд содержит не более 20 камней.

Формат выходного файла

Выведите на первой строке выходного файла "`Andrew`" или "`Mike`", в зависимости от того, кто выиграет при оптимальной игре обоих игроков. Если выиграет Андрюша, то на второй строке выведите номер ряда, в котором ему следует сделать первый ход, и номер камня в этом ряду, который следует забрать.

Пример

<table>
<tr><th>pebbles.in</th><th>pebbles.out</th></tr>
<tr><td>6
BBBRRB
RRRBBR
RBRBR
RB
BRBRRRR
BRBRB</td><td>Andrew
5 3</td></tr>
</table>

Задача G. Проблемы принцессы

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

В некотором царстве, в некотором государстве, жила была принцесса. Принцесса была настолько красива, что каждый юноша в том государстве хотел бы жениться на ней. Но принцесса была очень привередлива, так что она бы никогда не согласилась выйти замуж за кого угодно. Она твердо решила выйти замуж только за самого красивого юношу в государстве.

Но перед принцессой встала небольшая проблема — как же выбрать самого красивого? Конечно, если бы все юноши собрались одновременно и выстроились перед принцессой, она немедленно выбрала бы самого красивого, но молодые люди обычно заняты полезными делами — играют в интеллектуальные игры, сражаются на шпагах, бегают за девушками. Конечно, принцесса нашла решение — она будет вызывать по одному юноше в день, и таким образом выберет, за кого она выйдет замуж.

К счастью, у принцессы очень хорошая память. После того, как она хотя бы на секунду взглянет на юношу, она всегда может сказать про него и любого другого, какой из них более красив. У принцессы настолько тонкий вкус, что никакие два юноши не кажутся ей одинаково красивыми.

Принцесса составила список всех n красивых юношей в царстве и упорядочила их в случайном порядке. Теперь она планирует вызывать их по одному в день и действовать следующим образом: после того, как к ней приходит очередной юноша, она принимает решение. Либо она выходит за него замуж, либо нет. После того как она выйдет замуж за какого-либо юношу, она продолжит вызывать остальных юношей, чтобы убедиться, что она вышла замуж за самого красивого. Если это действительно так, то они живут долго и счастливо. В противном случае (если она либо вышла замуж на не самого красивого, либо не вышла замуж вообще), принцесса покончит жизнь самоубийством.

Помогите принцессе разработать стратегию, которая максимизирует вероятность того, что она выйдет замуж за наиболее красивого юношу. Единственное, что принцесса может делать, чтобы принять решение, выходить ли замуж за очередного юношу — это сравнить его красоту со всеми предыдущими.

Формат входного файла

Входной файл содержит число n ($1 \leq n \leq 2006$).

Формат выходного файла

Выведите в выходной файл одно число — вероятность, что принцесса выйдет замуж за самого красивого юношу при использовании оптимальной стратегии.

Ваш ответ должен отличаться от правильного не более чем на 10^{-8}.

Пример

princess.in	princess.out
1	1.0
2	0.5
3	0.5

В последнем примере принцесса должна действовать следующим образом. За первого юношу выходить замуж не следует. За второго юношу надо выходить замуж в том и только в том случае, если он красивее первого. Аналогично, за третьего юношу надо выходить замуж если он красивее обоих предыдущих.

При использовании этой стратегии принцесса не выйдет замуж за самого симпатичного в случае, если юноши будут упорядочены по красоте следующим образом: $\langle 1, 2, 3\rangle$, $\langle 3, 1, 2\rangle$, $\langle 3, 2, 1\rangle$, во всех остальных случаях принцесса выйдет замуж за самого красивого.

Задача H. Через речку

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

Дима и Федя пошли на прогулку. Они вышли из Диминого дома, расположенного у западного конца длинной прямой дороги, и дошли до Фединого дома, расположенного у восточного конца этой дороги. Во время прогулки они заметили, что перешли по разным мостам через речку Извилистую n раз.

К сожалению, никто из ребят не помнит, как именно течет речка. Например, Дима утверждает, что она течет с юга на север, а Федя — с севера на юг. Они уже собирались подраться, когда пришел Федин папа. Выслушав аргументы ребят, он предложил им посчитать — а сколько возможно различных конфигураций речки относительно дороги.

Но, кажется, они не справляются. Требуется ваша помощь!

Формат входного файла

Входной файл содержит число n ($1 \le n \le 16$).

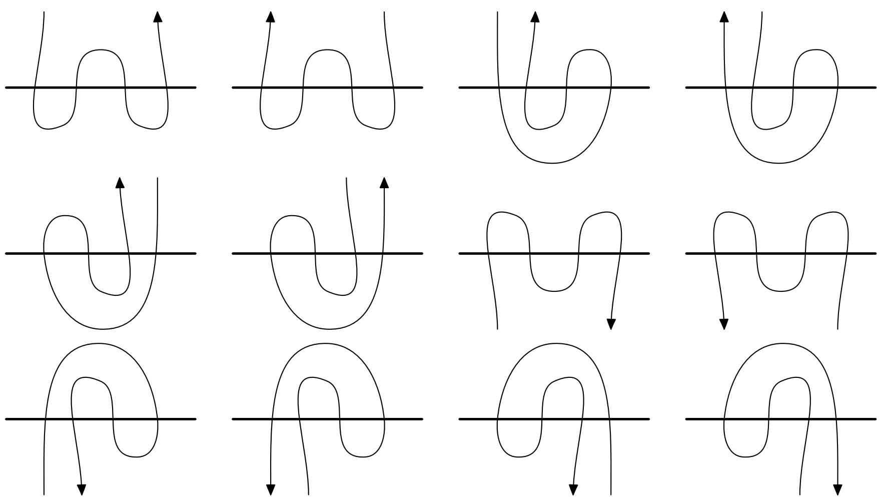
Формат выходного файла

Выведите одно число — количество возможных конфигураций речки.

Пример

river.in	river.out
1	2
2	4
3	4
4	12

Все возможные конфигурации речки в последнем примере приведены на рисунке.



Задача I. Range Minimum Query

Имя входного файла:	`rmq.in`
Имя выходного файла:	`rmq.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайт

Компания *Giggle* открывает новый офис в России, и вы решили пройти интервью, чтобы устроится в нее на работу. Вам предложили в качестве тестового задания реализовать специальную структуру данных.

Структура данных поддерживает список целых чисел, пронумерованных по порядку, начиная с единицы. Исходно список пуст. Ваша структура данных должна поддерживать следующие две операции:

- запрос: "`? i j`" — требуется вернуть минимальное число между i-м и j-м по номеру в списке, включительно;
- изменение: "`+ i x`" — добавить элемент x после i-го элемента списка. Если $i = 0$, то элемент надо добавить в начало списка.

Конечно, ваша структура данных должна быть достаточно производительной.

Формат входного файла

Первая строка входного файла содержит число n — количество операций, которые надо выполнить ($1 \le n \le 200\,000$). Следующие n строк описывают операции. Можете считать, что все запросы корректны, то есть не происходит "выходов за границы списка". Числа, которые добавляются в список, не превышают по модулю 10^9.

Формат выходного файла

Для каждого запроса выведите в выходной файл искомое минимальное число.

Пример

rmq.in	rmq.out
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Приведенная таблица показывает, как изменяется список, приведенный в примере, по мере добавления в него элементов.

Операция	**Список после операции**
исходно	*пуст*
`+ 0 5`	5
`+ 1 3`	5, 3
`+ 1 4`	5, 4, 3
`+ 0 2`	2, 5, 4, 3
`+ 4 1`	2, 5, 4, 3, 1