Задача A. Шары (Division 1 Only!)

Разработчик:	Антон Майдель
Имя входного файла:	`balls.in`
Имя выходного файла:	`balls.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Поиск в ширину, как и скрипты на PHP, может работать до десяти раз дольше.

На прямоугольной доске ширины N клеток и высоты M клеток установлено от двух до четырёх шаров. Каждый шар занимает отдельную клетку и покрашен либо в чёрный, либо в белый цвет. В любой момент времени в одной клетке может быть не более одного шара. Клетки, на которых не стоят шары, могут быть свободными, а могут быть заняты стеной. За один ход можно толкнуть любой шар в одном из четырёх направлений: вниз, вверх, влево или вправо. После толчка шар покатится и остановится либо по достижении края доски, либо уперевшись в другой шар или стену.

Требуется, выполнив минимальное число толчков, расположить шары согласно заданному шаблону. Для соответствия позиции на доске шаблону нужно, чтобы существовал такой параллельный перенос шаблона, после которого положения и цвета всех шаров в шаблоне и на доске будут совпадать. Поворачивать и отражать шаблон нельзя. Часть шаблона, не содержащая шаров, может выходить за пределы доски.

Формат входных данных

Входные данные состоят из одного теста. Первая строка ввода содержит два целых числа M и N — высоту и ширину доски ($5 \leqslant M, N \leqslant 8$). Далее в M строках, каждая из которых содержит ровно N символов, заданы клетки доски. Стенки отмечены решёткой (‘`#`’), свободные поля — точками (‘`.`’), чёрные шары — цифрой ‘`0`’, а белые шары — цифрой ‘`1`’. Общее количество шаров на доске не меньше двух и не больше четырёх.

Следующая строка содержит два целых числа H и W — высоту и ширину шаблона ($1 \leqslant H, W \leqslant 5$). Далее в H строках, каждая из которых содержит ровно W символов, задан шаблон. Цвета шаров в шаблоне заданы также цифрами ‘`0`’ и ‘`1`’, остальные символы шаблона — всегда звёздочки (‘`*`’). Каждая звёздочка может соответствовать пустой клетке, клетке со стеной или клетке за пределами доски. Количество и тип шаров в шаблоне и на доске совпадают. Гарантируется, что начальная позиция не соответствует шаблону.

Формат выходных данных

В первой строке выведите минимальное количество ходов, необходимых для решения поставленной задачи. В оставшихся строках выведите описание ходов. Описание хода состоит из положения (номер строки и номер столбца) толкаемого шара и направления толчка. Строки нумеруются с единицы сверху вниз, а столбцы — с единицы слева направо. Направление записывается следующим образом: ‘`D`’ — толчок вниз, ‘`U`’ — вверх, ‘`L`’ — влево, ‘`R`’ — вправо. Гарантируется, что во всех тестах, на которых будет проверяться ваша программа, решение поставленной задачи существует.

Примеры

balls.in	balls.out
5 6 .....1 .####. ....#. .####. 0....0 2 2 00 1*	8 5 1 U 1 1 R 5 6 L 5 1 U 1 5 L 1 6 D 5 6 L 5 1 U
5 5 ....0 #...1 .... 3 4 **** *0** **1*	2 1 5 L 2 5 L

Задача B. Мосты в дереве (Division 1 Only!)

Разработчик:	Сергей Копелиович
Имя входного файла:	`bridges2.in`
Имя выходного файла:	`bridges2.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Уже декабрь, скоро... Скоро сессия!

Дед Мороз

Мальчик Серёжа почти закончил университет. Осталось только сдать последнюю сессию и защитить диплом. Сессия, прямо скажем, — это не проблема: Серёжа уже сдавал её в прошлом году, а вот диплом — задача непростая.

Серёжа пару лет назад уже выбрал тему диплома: это «Dynamic 2-Connectivity Problem». В такой задаче присутствует динамически меняющийся граф, и нужно быстро отвечать на запрос «количество мостов в графе в текущий момент времени».

Серёжа придумал несколько решений. В одном из них, наиболее простом для реализации, возникла одна несколько неприятная подзадача. Серёжа написал код, но тот оказался сложным и длинным...

И тут ему стало интересно, а вдруг есть более простая реализация? Более короткая и не менее быстрая. Для скорейшего получения ответа на столь любопытный вопрос решено было дать задачу на ближайший чемпионат родного университета.

До предзащиты осталось всего несколько месяцев, и Серёже придется реализовать ещё много забавных алгоритмов. Времени все меньше, а если он не успеет, то не защитится, и ему придется восстанавливаться на пятый курс. Опять.

Помогите, пожалуйста, Серёже в нелёгком деле написания диплома, и попробуйте сразиться с этой неприятной подзадачей.

Напомним, что *дерево* — неориентированный связный граф без циклов. *Мостом* называется ребро неориентированного графа, при удалении которого количество компонент связности этого графа увеличивается.

Дано дерево `YourTree` из N ($1 \leqslant N \leqslant 100\,000$) вершин. Ваша задача — подсчитывать количество мостов в графах, полученных добавлением к дереву `YourTree` наборов из K_i рёбер.

Формат входных данных

В первой строке ввода записаны целые числа N и M ($1 \leqslant N \leqslant 100\,000$, $1 \leqslant M \leqslant 100\,000$) — количество вершин в дереве и количество запросов.

Во второй строке находится описание дерева `YourTree`: последовательность из $N-1$ целого числа p_2, p_3, ..., p_N, задающая рёбра $(2, p_2)$, $(3, p_3)$, ..., (N, p_N). Соседние числа разделены пробелами. Гарантируется, что $p_i < i$, а также что эти рёбра образуют дерево.

Далее идут описания M запросов, по одному на строке. Запрос с номером i описывается неотрицательным целым числом K_i и K_i парами чисел от 1 до N — рёбрами, которые добавляются к дереву при построении графа.

Сумма K_i по всем запросам не превосходит 100 000.

Заметим, что в полученных графах допустимы петли и кратные рёбра. Помните, что кратные рёбра не могут являться мостами.

Формат выходных данных

В ответ на каждый запрос выведите одно целое число — количество мостов в графе, полученном добавлением к дереву `YourTree` заданного набора рёбер.

Пример

bridges2.in	bridges2.out
7 8 1 1 2 2 3 3 1 4 5 3 4 5 6 7 3 2 1 5 6 1 1 1 1 3 6 2 4 3 2 7 1 5 1 3 1 2 1 3 1 6	4 0 2 6 5 2 4 3

Задача C. Гладкие числа (Division 1 Only!)

Разработчик:	Сергей Копелиович
Имя входного файла:	`bsmooth.in`
Имя выходного файла:	`bsmooth.out`
Ограничение по времени:	4 секунды (*10 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Гладкие и пушистые...

Учебник по теории чисел для школьников

Натуральное число называется B-*гладким*, если все его простые делители не превосходят B.

Напишите программу, которая по числам B и N найдёт количество B-*гладких* чисел, не превосходящих N.

Формат входных данных

В единственной строке ввода записано два целых числа N и B ($1 \leqslant N \leqslant 8 \cdot 10^{18}$, $2 \leqslant B \leqslant 100$).

Формат выходных данных

Выведите одно целое число: количество B-*гладких* чисел, не превосходящих N.

Пример

bsmooth.in	bsmooth.out
1000 10	141

Задача D. Карточный домик (Division 1 Only!)

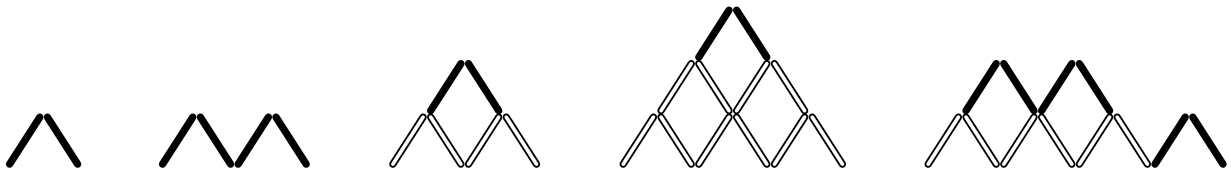
Разработчик:	Ольга Бурсиан
Имя входного файла:	`cardhouse.in`
Имя выходного файла:	`cardhouse.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Ёжик хочет построить карточный домик к тому моменту, как к нему придёт в гости Медвежонок. Этот домик строится в соответствии со следующими рекурсивными правилами.

1. Две карты, стоящие наклонно, соприкасаясь верхними краями и образуя угол, называются *уголком*.
2. Уголок является карточным домиком.
3. Два карточных домика, стоящие рядом бок о бок, тоже являются карточным домиком.
4. Если два уголка, принадлежащие некоторому карточному домику, стоят рядом бок о бок на одной высоте, то, поставив на них сверху уголок так, чтобы он нижними концами своих карт опирался на верхние концы этих уголков, мы тоже получим карточный домик.

Уголок карточного домика, над которым нет другого уголка, называется *пиком* карточного домика.

На рисунке ниже представлены несколько примеров карточных домиков. Более тёмные уголки являются пиками.



У Ёжика есть $2N$ карт. Он хочет построить карточный домик изо всех своих карт так, чтобы число пиков было минимальным. Найдите это минимальное число, чтобы помочь ему.

Формат входных данных

Входной файл состоит из одного целого числа N ($0 \leqslant N \leqslant 1\,000\,000$).

Формат выходных данных

Выведите минимальное число пиков карточного домика, который может быть построен ровно из $2N$ карт.

Примеры

cardhouse.in	cardhouse.out
3	1
4	2

Задача E. Рыцари и лжецы (Division 1 Only!)

Разработчик:	Иван Казменко
Имя входного файла:	`knl.in`
Имя выходного файла:	`knl.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

На острове Надежды живут n человек, каждый из которых — либо рыцарь, который всегда говорит правду, либо лжец, высказывания которого всегда неверны. Каждый житель острова имеет уникальный номер — целое число от 1 до n, включительно.

Ежегодно жители собираются в центре острова, после чего каждый говорит ровно одну фразу вида «житель с таким-то номером — рыцарь» или «житель с таким-то номером — лжец».

В этом году Вася приплыл на остров к началу собрания. Он услышал и записал несколько высказываний, после чего на собрании был объявлен перерыв на обед. Сейчас Вася хочет промоделировать дальнейшее развитие событий.

Назовём *протоколом* собрания спикок из n высказываний: что сказал первый островитянин, что сказал второй, что — третий, ..., а что — n-й островитянин. В Васиной модели каждый из ещё не высказавшихся островитян скажет одно из $2n$ возможных высказываний равновероятно и независимо от того, что скажут другие. Вот эти возможные высказывания: «первый житель — рыцарь», «первый житель — лжец», «второй житель — рыцарь», ..., «n-й житель — лжец». После этого по полученному протоколу нужно найти *решение* — для каждого жителя определить, рыцарь он или лжец, так, чтобы все высказывания из протокола были верны.

Заметим, что для разных протоколов количество возможных решений различно. В частности, существуют протоколы, для которых нет ни одного решения: к примеру, если какой-то островитянин сказал про себя самого, что он — лжец, решений не существует независимо от того, что и про кого сказали остальные. Напротив, если получится протокол, в котором каждый островитянин сказал про себя самого, что он рыцарь, каждый житель может независимо от остальных быть как рыцарем, так и лжецом, и возможных решений оказывается 2^n.

Теперь Вася просит вас оценить, каковы его шансы получить решение. Рассмотрим момент, когда протокол уже зафиксирован, то есть недостающие высказывания выбраны независимо и равновероятно, и пора приступать к поиску решения. Васю интересуют две величины. Первая — это математическое ожидание количества решений. Вторая — это вероятность того, что хотя бы одно решение существует.

К сожалению, записывая высказывания островитян, Вася мог допустить ошибку. Поэтому исходные данные могут быть противоречивы, и из-за этого может оказаться, что при любом возможном протоколе количество решений равно нулю. В этом случае две искомые величины следует также принять равными нулю.

Напомним, что математическое ожидание случайной величины — это сумма по всем возможным исходам (в нашем случае — протоколам) вероятности такого исхода, умноженной на значение этой величины (в нашем случае — количество решений) при этом исходе.

Формат входных данных

В первой строке ввода заданы через пробел два целых числа n и k — количество жителей острова и количество услышанных Васей высказываний ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant n$). В следующих k строках записаны высказывания. Каждая из этих строк имеет вид i j s, где i — номер островитянина, которому принадлежит это высказывание ($1 \leqslant i \leqslant n$), j — номер остро-

витянина, к которому относится высказывание ($1 \leqslant j \leqslant n$), а s — либо строка «knight», либо строка «liar» в зависимости от того, считает ли островитянин i островитянина j рыцарем или лжецом.

Гарантируется, что все высказывания сделаны разными жителями острова. Помните, что входные данные могут оказаться противоречивыми.

Формат выходных данных

В первой строке вывода запишите через пробел два числа: математическое ожидание количества решений и вероятность того, что хотя бы одно решение существует. Ответы считаются верными, если они отличаются от точных не более чем на 10^{-9}.

Пример

knl.in	knl.out
3 2 1 2 liar 3 1 knight	1.0000000000 0.5000000000

Пояснение к примеру

В этом примере житель 1 говорит, что житель 2 — лжец, а житель 3 считает, что житель 1 — рыцарь. Рассмотрим все шесть возможных итоговых протоколов. Они отличаются только высказываниями второго жителя. Каждый из этих протоколов имеет вероятность $\frac{1}{6}$. В скобках указано количество различных решений для соответствующего итогового протокола. Сами решения закодированы строкой из трёх букв, описывающих трёх жителей по порядку (Р — рыцарь, Л — лжец).

1. Житель 2 говорит, что житель 1 — рыцарь (0 решений).
2. Житель 2 говорит, что житель 1 — лжец (2 решения: РЛР и ЛРЛ).
3. Житель 2 говорит, что житель 2 — рыцарь (2 решения: РЛР и ЛРЛ).
4. Житель 2 говорит, что житель 2 — лжец (0 решений).
5. Житель 2 говорит, что житель 3 — рыцарь (0 решений).
6. Житель 2 говорит, что житель 3 — лжец (2 решения: РЛР и ЛРЛ).

Математическое ожидание количества решений равно $\frac{1}{6}\cdot 0+\frac{1}{6}\cdot 2+\frac{1}{6}\cdot 2+\frac{1}{6}\cdot 0+\frac{1}{6}\cdot 0+\frac{1}{6}\cdot 2 = 1$. Вероятность того, что хотя бы одно решение существует, равна $0+\frac{1}{6}+\frac{1}{6}+0+0+\frac{1}{6}=\frac{1}{2}$.

Задача F. Наибольшая общая подпоследовательность (Division 1 Only!)

Разработчик:	Антон Майдель
Имя входного файла:	`lcs.in`
Имя выходного файла:	`lcs.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

> Видали мы такие бояны, рядом с которыми этот выглядит, как фортепьяно.

Научно-исследовательский институт данных строк (НИИДС) нуждается в программе для поиска наибольшей общей подпоследовательности двух массивов специального вида. Длина каждого массива равна $2N$. Оба массива состоят из целых чисел от 1 до N, причём каждое число встречается ровно два раза в первом массиве и ровно два раза во втором.

Вася вызвался написать такую программу, но эта задача оказалась ему не под силу. Помогите Васе!

Формат входных данных

Входные данные состоят из одного или более тестов.

Каждый тест состоит из трёх строк. В первой из них задано целое число N ($1 \leqslant N \leqslant 50\,000$). Во второй строке заданы через пробел $2N$ целых чисел — первый массив. В третьей строке заданы через пробел $2N$ целых чисел — второй массив. Гарантируется, что в каждом из массивов каждое из чисел 1, 2, ..., N встречается ровно два раза.

Общее количество тестов не превосходит 1000. Сумма всех значений N во вводе не превосходит 100 000.

Входные данные завершаются строкой, состоящей из числа 0.

Формат выходных данных

В ответ на каждый тест выведите две строки. В первой строке выведите M — длину наибольшей общей подпоследовательности. Во второй строке выведите через пробел M чисел — саму подпоследовательность. Если оптимальных ответов несколько, выведите любой из них.

Пример

lcs.in	lcs.out
2 1 2 1 2 2 1 2 1 3 1 1 2 2 3 3 3 3 2 2 1 1 0	3 1 2 1 2 1 1

Задача G. Игра «Жизнь» (Division 1 Only!)

Разработчик:	Сергей Копелиович
Имя входного файла:	`life.in`
Имя выходного файла:	`life.out`
Ограничение по времени:	7 секунд (*8 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Что наша игра? Жизнь!

Дана клетчатая доска $W \times H$. Каждая клетка исходно является либо чёрной, либо белой. Клетки называются соседними, если у них есть общая сторона.

Повторяется следующая процедура:

- Находится белая клетка, у которой не менее двух чёрных соседей. Если такой клетки нет, процесс завершается.
- Найденная клетка перекрашивается в чёрный цвет.

Напишите программу, которая вычислит количество различных раскрасок доски, которые могут получиться по завершении процесса.

Поскольку ответ может быть большим, достаточно найти его остаток по модулю 10^{18}.

Формат входных данных

В первой строке ввода записаны размеры доски: два целых числа W и H ($1 \leqslant W, H \leqslant 13$). В следующих H строках записано по W символов, описывающих саму доску:

- Символ «`B`» обозначает, что клетка в начальной раскраске была чёрной.
- Символ «`.`» обозначает, что исходный цвет клетки неизвестен, то есть он может быть как чёрным, так и белым.

Формат выходных данных

Выведите одно целое число: остаток количества различных возможных конечных раскрасок по модулю 10^{18}.

Пример

life.in	life.out
3 4 B.B .B.	3

Пояснение к примеру

В данном примере возможны следующие конечные раскраски (здесь символом «`W`» обозначаются белые клетки):

```
BBB   BBB   BBB
BBB   BBB   BBB
WWW   BBB   BBB
WWW   WWW   BBB
```

Задача H. Memcached (Division 1 Only!)

Разработчик:	Антон Майдель
Имя входного файла:	`memcached.in`
Имя выходного файла:	`memcached.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Тигры не любят бурундуков.

Из неопубликованной книги о Винни-Пухе

Научно-исследовательский институт данных строк (НИИДС) поручил Васе реализовать подмножество протокола *memcached*. В этом подмножестве есть всего две команды: `SET` и `GET`. Команда `SET` используется для присваивания переменной некоторого значения, а команда `GET` — для вывода значения указанной переменной или определения того, что эта переменная ещё не была инициализирована.

Помогите Васе! Синтаксис команд и формат ответов на них подробно описаны ниже.

Формат входных данных

Входные данные состоят из не более чем 100 строк, каждая строка — из не более чем 100 символов, причём все встречающиеся в строках символы имеют ASCII-коды от 32 до 126. *Имя переменной* — строка, состоящая из непробельных символов, длина которой лежит в пределах от 1 до 50 символов. Имена переменных, отличающиеся регистром букв, следует считать различными. Переменная считается *инициализированной*, если для неё хотя бы один раз вызвана команда `SET`.

Команда `SET` занимает две строки ввода:

```
SET <имя переменной> 0 0 <длина значения>
<значение переменной>
```

Длина второй строки всегда равна длине значения, указанной в первой строке.

Команда `GET` занимает одну строку ввода:

```
GET <имя переменной>
```

Первые строки команд `GET` и `SET` не содержат двух и более пробелов подряд. Названия команд всегда состоят из заглавных букв. Гарантируется, что на входе содержатся только корректные команды, а также что между любыми двумя соседними командами нет пустых строк.

Формат выходных данных

В ответ на команду `SET` нужно вывести одну строку «`STORED`» и запомнить значение указанной переменной.

В ответ на команду `GET` для инициализированной переменной нужно вывести три строки:

```
VALUE <имя переменной> 0 <длина значения>
<значение переменной>
END
```

В ответ на команду GET для неинициализированной переменной нужно вывести одну строку «END».

Пример

memcached.in	memcached.out
GET empty SET puzzle 0 0 25 Do kittens eat chipmunks? GET puzzle SET puzzle 0 0 33 Shhhhhhhhhh, I'm trying thinking. GET puzzle	END STORED VALUE puzzle 0 25 Do kittens eat chipmunks? END STORED VALUE puzzle 0 33 Shhhhhhhhhh, I'm trying thinking. END

Задача I. Найдите путь (Division 1 Only!)

Разработчик:	Наталья Гинзбург
Имя входного файла:	`pathfind.in`
Имя выходного файла:	`pathfind.out`
Ограничение по времени:	3 секунды (*5 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Дан неориентированный взвешенный граф без петель и кратных рёбер. Также дана последовательность чисел. Выведите путь в графе, длины рёбер в котором образуют заданную последовательность. Если таких путей несколько, выведите лексикографически минимальный. Если же таких путей не существует, выведите «`No solution`» (без кавычек).

Обратите внимание, что число в последовательности может не соответствовать ни одному ребру заданного графа.

Пути в графе задаются как последовательности номеров вершин графа. Любые две соседние вершины в пути должны быть соединены ребром.

Путь a лексикографически меньше пути b той же длины тогда и только тогда, когда в первой позиции, в которой пути различаются, номер вершины в a меньше номера вершины в b.

Формат входных данных

В первой строке ввода заданы через пробел три целых числа n, m и k — количество вершин графа, рёбер графа и количество рёбер в искомом пути соответственно ($2 \leqslant n \leqslant 500$, $1 \leqslant m \leqslant \frac{n(n-1)}{2}$, $1 \leqslant k \leqslant 500$).

В следующих m строках описаны рёбра графа: в каждой строке заданы через пробел три целых числа u, v и l — номера двух смежных вершин и длина соединяющего их ребра ($1 \leqslant u, v \leqslant n$, $1 \leqslant l \leqslant 10^9$).

В следующей строке заданы через пробел k целых чисел. Эти числа также лежат в пределах от 1 до 10^9.

Формат выходных данных

Если искомого пути не существует, выведите строку «`No solution`». В противном случае в первой строке выведите искомый путь: последовательность из $k+1$ целого числа, состоящую из номеров первой, второй, ..., $(k+1)$-ой вершин пути. Вершины графа нумеруются с единицы. Выведенный путь должен быть лексикографически минимальным. Соседние числа в последовательности следует разделять пробелами.

Примеры

pathfind.in	pathfind.out
3 3 3 1 2 10 2 3 20 1 3 30 30 10 20	3 1 2 3
3 3 3 1 2 10 2 3 20 1 3 30 30 10 30	No solution
3 3 3 1 2 10 2 3 20 1 3 30 30 10 40	No solution
4 5 6 1 2 10 2 3 10 1 3 10 1 4 10 2 4 10 10 10 10 10 10 10	1 2 1 2 1 2 1