Задача A. a.*b

В задачах 1-4 решением является текстовый файл, содержащий описание детерминированного конечного автомата в установленном формате.

В первой задаче вам необходимо реализовать автомат, принимающий все слова, начинающиеся с буквы a и заканчивающиеся буквой b (то есть удовлетворяющие регулярному выражению `a.*b`).

Формат описания автомата

Автомат обрабатывает текстовые строки, состоящие из произвольных ASCII-символов (без пробелов и специальных символов). Автомат имеет N состояний, пронумерованных от 1 до N, состояние номер 1 является начальным, у автомата есть одно или несколько терминальных состояний.

Первая строка описания автомата содержит число N. Вторая строка описания автомата содержит номера терминальных состояний автомата (числа от 1 до N), разделенных пробелами.

Все следующие строки описания автомата задают переходы. Описание перехода состоит из 3 частей, записанных через пробел. Первая часть — номер начального состояния автомата. Вторая часть: последовательность из символов входных данных, по которым осуществляется переход. Третья часть: номер конечного состояния. Например, запись:

```
2 abc 3
```

означает, что если автомат находится в состоянии 2 и получает на вход один из символов “a”, “b”, “c”, то он переходит в состояние 3. Для каждой пары (состояние, символ) может быть задано не более одного перехода из данного состояния по данному символу.

Вместо второй части может присутствовать символ “*”, обозначающий переход по любому символу, для которого переход не задан явно.

Автомат принимает входное слово, если он завершает работу в терминальном состоянии. Автомат не принимает слово, если он останавливается в нетерминальном состоянии, или если для текущего состояния автомата и входного символа нет перехода (в том числе заданного при помощи “*”) в описаннии автомата.

Пример автомата, принимающего все слова, содержащие четное число гласных букв:

```
2
1
1 aeiou 2
1 * 1
2 aeiou 1
2 * 2
```

Задача B. Делимость на 3

Автомат получает на вход целое неотрицательное число (последовательность десятичных цифр). Реализуйте автомат, который принимает входное слово, если число делится на 3. Например, автомат должен принимать слова “0”, “39”, “123” и не должен принимать слова “7”, “14”, “179” и т.д.

Задача C. Каких букв больше?

Реализуйте автомат, распознающий слова, в которых число букв “a” больше, чем число букв “b”. Дополнительное условие: все входные слова содержат не более 10 букв. Слово может содержать любые латинские буквы.

Задача D. Сложное выражение

Реализуйте автомат, распознающий регулярное выражение:

```
(ab{1,2}|c)+(b*c|b+a*|c+a+)
```

Задача E. Без комментариев

Имя входного файла:	`comments.in`
Имя выходного файла:	`comments.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Издревле почти в каждом монастыре ведутся летописи событий происходящих внутри и за пределами самого монастыря. Не исключением является и Монастырь Светлой Луны. Все свои наблюдения монахи тщательно записывали в особые дневники (Даарны). Как часто случается, в этих летописях встречается не только описание реальных событий, но и комментарии самого летописца. К счастью, в Монастыре Светлой Луны был заведен порядок, что комментарии должны отделяться от описания событий одним из следующих способов:

- Комментарий начинается с «//» и продолжается до конца данной строки (символ перевода строки не является частью комментария).
- Комментарий начинается с «{» и продолжается до ближайшего вхождения «}».
- Комментарий начинается с «/*» и продолжается до ближайшего вхождения «*/».

Внутри комментария могут встречаться любые символы. Известно, что монахи никогда не ошибаются и не оставляют комментарии незакрытыми. Также известно, что после удаления комментариев в тексте не возникнут новые комментарии.

По совету Наставника монахи хотят переписать все летописи, убрав из него все комментарии. Ваша цель – помочь им в этом нелегком деле.

Формат входных данных

Во входном файле содержится летопись длиной не более 10^6 символов. Каждая строка летописи не длиннее 250 символов.

Формат выходных данных

Выведите летопись, очищенную от комментариев.

Примеры

comments.in
When I find myself in times of trouble Mother Mary comes to me Speaking words of wisdom, "Let it be". // Original: http://en.lyrsense.com/beatles/let_it_be { Copyright: http://lyrsense.com }

comments.out
When I find myself in times of trouble Mother Mary comes to me Speaking words of wisdom, "Let it be".

Задача F. Количество слов

Имя входного файла:	`numwords.in`
Имя выходного файла:	`numwords.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Дан детерминированный конечный автомат. Определите, сколько существует различных слов длины K, принимаемых данным автоматом.

Формат входных данных

Первая строка входных данных содержит два целых числа N и M ($1 \leqslant N \leqslant 30$) — количество состояний в автомате и количество переходов. В следующих M строках записаны переходы данного автомата. Каждый переход задается тройкой S_i, C_i, T_i, где S_i — номер исходного состояния перехода ($1 \leqslant S_i \leqslant N$), C_i — символ, по которому осуществляется переход (строчная буква латинского алфавита), T_i — конечное состояние перехода ($1 \leqslant T_i \leqslant N$).

Далее записано число T — количество терминальных состяний автомата ($0 \leqslant T \leqslant N$). В следующей строке записано T различных чисел — номера терминальных состояний. Последняя строка входных данных содержит число K ($0 \leqslant K \leqslant 1000$) — длина входного слова.

Начальное состояние автомата имеет номер 1. Если в процессе работы автомата появится невозможный переход (то есть возникает комбинация состояния и символа, не описанная в списке возможных переходов), то такое входное слово считается не распознанным автоматом.

Формат выходных данных

Выведите остаток от деления числа всевозможных входных слов длины K, распознаваемых данным автоматом, на $10^9 + 7$.

Примеры

numwords.in	numwords.out
5 8 1 a 2 1 b 3 2 a 4 2 b 3 4 b 3 3 a 2 3 b 5 5 a 2 4 2 3 4 5 4	10

Задача G. Путевые строки

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

Рассмотрим ориентированный граф G, имеющий n вершин, пронумерованных натуральными числами от 1 до n. В графе G возможно наличие нескольких дуг между одной и той же парой вершин, а также дуг, ведущих из вершины в нее саму. Каждая дуга графа помечена некоторой буквой латинского алфавита. Каждому пути в графе G можно поставить в соответствии строку, состоящую из букв, написанных на последовательно проходимых в соответствии с этим путем дугах. Эта строка называется путевой меткой пути. Назовем строку S путевой строкой графа G, если в нем существует путь, путевая метка которого равна S.

Ваша задача посчитать остаток от деления на 1 000 000 количества путевых строк графа G, состоящих ровно из L символов.

Формат входных данных

В первой строке входных данных записаны целые числа n, m, L ($1 \leqslant n \leqslant 10$, $1 \leqslant m \leqslant 10\,000$, $1 \leqslant L \leqslant 100$), равные количеству вершин и ребер графа G, а также длине путевых строк, которые нужно искать. Следующие m строк задают дуги графа G. Каждая из этих строк содержит два натуральных числа a, b ($1 \leqslant a, b \leqslant n$) и маленькую латинскую букву c, что означает наличие дуги из вершины a в вершину b, помеченной символом c. Элементы каждой строки отделены друг от друга пробелами.

Формат выходных данных

Единственная строка выходных данных должна содержать одно число, равное остатку от деления количества путевых строк длины L в графе G на 1 000 000.

Примеры

path-string.in	path-string.out
4 4 100 1 2 a 2 3 b 3 4 a 4 1 b	2

В следующих задачах сегодняшнего дня (кроме regpcms2) вы должны написать программу, которая будет выводить на экран одну строку (в задаче regnum — две строки) — искомое регулярное выражение. Не забывайте символы «$» и «$\hat{}$» в тех выражениях, которые должны ловить целую строку. Например, для задания регулярного выражения, распознающего строки, начинающиеся с символа обратных слэш «\» можно сдать следующую программу:

```
print("^\\\\\")
```

Задача H. Regif. Присванивание в if'ах

Программисты, переходящие с паскаля на C-подобные языки, часто делают ошибки, записывая в операторе if одиночное равенство, а не двойное: 'if (a=0)' вместо 'if (a==0)'. Напишите регулярное выражение, позволяющее отлавливать такие ошибки. А именно, оно должно срабатывать (т.е. находить совпадение) на строках кода, содержащих присваивание внутри условия if'а, и не срабатывать на коде, такой ошибки не содержащей.

Поскольку правильные скобочные последовательности не являются регулярным языком и довольно сложно задаются даже perl-compatible регулярными выражениями, от вас не требуется коректно обрабатывать записи вида

```
if (a==0) b=c;
```

(т.е. когда одиночное равенство не находится в if'е потому, что все скобки сбалансировались и условие if'а кончилось). Вместо этого считайте, что после условия if'а всегда идет либо открывающая фигурная скобка, либо конец строки.

Считайте, что ваше выражение должо будет обрабатывать одиночные строки кода, т.е. его использование будет проходить следующим образом: разбили код на строки и в каждой строке отдельно поискали совпадения с этим регулярным выражением. Естественно, считайте, что условие if'а всегда находится в одной строке.

Для упрощения задачи ваше реулярное выражение может считать, что в строках не встречается строковых констант и комментариев.

Задача I. Regprime. Проверяем на простоту

В этой задаче будем натуральное число N записывать N единицами подряд, например, число 4 будем записывать как '1111'

Выведите регулярное выражение, проверяющее, что данное число, записаное указанным образом, является составным. Выражение должно работать для

произвольных натуральных чисел, бóльших единицы.

Например, регулярное выражение '^(11){2,}$' правильно работает для записей чисел от 2 до 8, но дает неправильный результат для числа 9.

Задача J. Regnum. Конвертация чисел

Пусть у вас есть текст, в котором встречаются вещественные числа, записанные с десятичной точкой и, возможно, с экспоненциальной частью (например, '1.25', '1.25e10', '1.25e+10' или '1.25E-10'). Ваша задача - с помощью регулярных выражений нормализовать их, а именно:

- заменить в записи всех вещественных чисел десятичную точку на десятичную запятую,
- удалить все ведущие нули как у мантиссы, так и у экспоненциальной части (обратите внимание, что перед десятичной запятой все равно должен остаться как минимум один ноль).

А именно, вам надо написать регулярное выражение и строку замены, которые будут решать эту задачу. (В строке замены можно использовать ссылки на скобки в начальной строке в виде '\2'.)

Вы должны сдать программу, которая выводит две строки: регулярное выражение и строку замены.

Задача строго не определена: вы должны корректно обрабатывать разумные записи, но неразумные записи (например, '1.1e1e1.1e1') можете обрабатывать как угодно, их не будет в тестах. Также считайте, что целые числа, а также числа без десятичной точки нормализовать не надо.

Пример

тест	ответ
0.1+012.3e-09	0,1+12,3e-9

Задача K. RegPCMS2. Разбор таблицы результатов

Имя входного файла: regpcms2.in
Имя выходного файла: regpcms2.out

Во входном файле вам дана html-страничка - результаты некоторой олимпиады по программированию, сформированные системой PCMS2 (см. пример на страничке параллели.) Выведите в выходной файл информацию по всем *успешным* посылкам в этом контесте. Каждую посылку выводите на отдельной строке в следующем формате:

<название команды> <идентификатор задачи> <время в минутах> <количество штрафных посылок по этой задаче>

Задачи нумеруются заглавными латинскими буквами, начиная с A. Посылки сортируйте по времени сдачи, при равных временах - по названию команды. Названия команд выводите в той же кодировке, в которой они заданы во входном файле (т.е. просто выводите те же байты).

P.S. Напишите программы с использованием регулярных выражений для поиска нужных фрагментов текста.

Пример

Выставлен на страничке параллели.