В задачах сегодняшнего дня вы должны написать программу, которая будет выводить на экран одну строку (в задаче Regnum — две строки) — искомое регулярное выражение. Не забывайте символы $ и ^ в тех выражениях, которые должны ловить целую строку.

RegDiv3. Учимся делить на три!

В этой задаче разрешим в двоичной записи чисел присутствие ведущих нулей. Кроме того, разрешим записывать число 0 в двоичной записи как ϵ (то есть как пустую строку).

Таким образом, строки «101» и «000101» являются двоичными записями числа 5, а строки «000000» и «» — двоичными записями числа 0.

Выведите регулярное выражение, проверяющее, что данное число, записанное в двоичной записи, делится на 3.

Например, регулярное выражение '`^(0|11)*$`' правильно работает для записей двоичных чисел от 0 до 8, но дает неправильный результат для числа 9.

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`' 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

`0.1+012.3e-09`	`0,1+12,3e-9`

Regif. Присванивание в if'ах

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

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

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

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

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

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

Regstring. Проверяем строку

Выведите регулярное выражение, проверяющее, что данное строка является корре ной заключенной в кавычки строковой константой языка C. Считайте, что в строке встречается символов перевода строки.

Например, строки `"test test"` и `"test \" test"` являются корректными строковы константами, а `"test""test"` — нет.

Regprime. Проверяем на простоту

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

Выведите регулярное выражение, проверяющее, что данное число, записаное указанн образом, является составным. Выражение должно работать для произвольных натура ных чисел, бóльших единицы.

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

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.

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

Пример

Выставлен на страничке параллели A′+.