Отборочный-2 10-11

RFC1924_1

В этой задаче требуется преобразовать IPv6 адрес в компактную форму в соответствии с RFC1924. IPv6 адрес предстваляет из себя 16-байтное число которое записывается в шестнадцатиричной системе счисления и группы из 2х байт разделяются двоеточием. При этом если две и более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие. Незначащие старшие нули в группах могут быть опущены.
Таким образом длина записи IPv6 в класической форме занимает от 2 до 39 символов.
Примеры адресов:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080::8:800:200C:417A

В стандарте RFC1924 можно записать адрес с помощью не более 20 символов. Адрес рассматривается как 128-битное целое число, которое записывается в системе счисления с основанием 85 с использованием следующих символов (в порядке возрастания значений):

'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&', '(',
')', '*', '+', '-', ';', '<', '=', '>', '?', '@', '^', '_',
'`', '{', '|', '}', и '~'.

Например, адрес 1080:0:0:0:8:800:200C:417A это число 21932261930451111902915077091070067066 в десятичной системе счисления, или 4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51 в 85-ричной. Что дает запись 4)+k&C#VzJ4br>0wv%Yp
Преобразуйте адрес
64:ff9b::bc2c:3267

в формат RFC1924

RFC1924_1

В этой задаче требуется преобразовать IPv6 адрес в компактную форму в соответствии с RFC1924. IPv6 адрес предстваляет из себя 16-байтное число которое записывается в шестнадцатиричной системе счисления и группы из 2х байт разделяются двоеточием. При этом если две и более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие. Незначащие старшие нули в группах могут быть опущены.
Таким образом длина записи IPv6 в класической форме занимает от 2 до 39 символов.
Примеры адресов:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080::8:800:200C:417A

В стандарте RFC1924 можно записать адрес с помощью не более 20 символов. Адрес рассматривается как 128-битное целое число, которое записывается в системе счисления с основанием 85 с использованием следующих символов (в порядке возрастания значений):

'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&', '(',
')', '*', '+', '-', ';', '<', '=', '>', '?', '@', '^', '_',
'`', '{', '|', '}', и '~'.

Например, адрес 1080:0:0:0:8:800:200C:417A это число 21932261930451111902915077091070067066 в десятичной системе счисления, или 4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51 в 85-ричной. Что дает запись 4)+k&C#VzJ4br>0wv%Yp
Преобразуйте адрес
64:ff9b::bc2c:326e

в формат RFC1924

RFC1924_1

В этой задаче требуется преобразовать IPv6 адрес в компактную форму в соответствии с RFC1924. IPv6 адрес предстваляет из себя 16-байтное число которое записывается в шестнадцатиричной системе счисления и группы из 2х байт разделяются двоеточием. При этом если две и более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие. Незначащие старшие нули в группах могут быть опущены.
Таким образом длина записи IPv6 в класической форме занимает от 2 до 39 символов.
Примеры адресов:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080::8:800:200C:417A

В стандарте RFC1924 можно записать адрес с помощью не более 20 символов. Адрес рассматривается как 128-битное целое число, которое записывается в системе счисления с основанием 85 с использованием следующих символов (в порядке возрастания значений):

'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&', '(',
')', '*', '+', '-', ';', '<', '=', '>', '?', '@', '^', '_',
'`', '{', '|', '}', и '~'.

Например, адрес 1080:0:0:0:8:800:200C:417A это число 21932261930451111902915077091070067066 в десятичной системе счисления, или 4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51 в 85-ричной. Что дает запись 4)+k&C#VzJ4br>0wv%Yp
Преобразуйте адрес
2a02:6b8::feed:0ff

в формат RFC1924

RFC1924_1

В этой задаче требуется преобразовать IPv6 адрес в компактную форму в соответствии с RFC1924. IPv6 адрес предстваляет из себя 16-байтное число которое

записывается в шестнадцатиричной системе счисления и группы из 2х байт разделяются двоеточием. При этом если две и более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие. Незначащие старшие нули в группах могут быть опущены.
Таким образом длина записи IPv6 в класической форме занимает от 2 до 39 символов.
Примеры адресов:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080::8:800:200C:417A

В стандарте RFC1924 можно записать адрес с помощью не более 20 символов. Адрес рассматривается как 128-битное целое число, которое записывается в системе счисления с основанием 85 с использованием следующих символов (в порядке возрастания значений):

'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&', '(',
')', '*', '+', '-', ';', '<', '=', '>', '?', '@', '^', '_',
'`', '{', '|', '}', и '~'.

Например, адрес 1080:0:0:0:8:800:200C:417A это число 21932261930451111902915077091070067066 в десятичной системе счисления, или 4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51 в 85-ричной. Что дает запись 4)+k&C#VzJ4br>0wv%Yp
Преобразуйте адрес
2a02:6b8:0:1::feed:a11

в формат RFC1924

Пароль-2

Петя зашифровал свой буквенный пароль, чтобы никто другой не смог его прочитать, но потом понял, что даже сам не может этого сделать. Закодированный пароль представляет собой число, являющееся произведением кодов всех букв. Буквы кодируются подряд по своему расположению в английском алфавите, причем буква A кодируется двойкой, B – тройкой и т. д. Помогите Пете подобрать пароль, если им записано число 2348346. Отправьте текст, где выписаны всевозможные пароли в нижнем регистре через символ перевода строки в алфавитном порядке или пустой ответ, если по данному числу невозможно получить никакого слова. После ответа, опишите как он был получен.

Пароль-2

Петя зашифровал свой буквенный пароль, чтобы никто другой не смог его прочитать, но потом понял, что даже сам не может этого сделать. Закодированный пароль

представляет собой число, являющееся произведением кодов всех букв. Буквы кодируются подряд по своему расположению в английском алфавите, причем буква A кодируется двойкой, B – тройкой и т. д. Помогите Пете подобрать пароль, если им записано число 1677390. Отправьте текст, где выписаны всевозможные пароли в нижнем регистре через символ перевода строки в алфавитном порядке или пустой ответ, если по данному числу невозможно получить никакого слова. После ответа, опишите как он был получен.

Пароль-2

Петя зашифровал свой буквенный пароль, чтобы никто другой не смог его прочитать, но потом понял, что даже сам не может этого сделать. Закодированный пароль представляет собой число, являющееся произведением кодов всех букв. Буквы кодируются подряд по своему расположению в английском алфавите, причем буква A кодируется двойкой, B – тройкой и т. д. Помогите Пете подобрать пароль, если им записано число 3913910. Отправьте текст, где выписаны всевозможные пароли в нижнем регистре через символ перевода строки в алфавитном порядке или пустой ответ, если по данному числу невозможно получить никакого слова. После ответа, опишите как он был получен.

Пароль-2

Петя зашифровал свой буквенный пароль, чтобы никто другой не смог его прочитать, но потом понял, что даже сам не может этого сделать. Закодированный пароль представляет собой число, являющееся произведением кодов всех букв. Буквы кодируются подряд по своему расположению в английском алфавите, причем буква A кодируется двойкой, B – тройкой и т. д. Помогите Пете подобрать пароль, если им записано число 1118260. Отправьте текст, где выписаны всевозможные пароли в нижнем регистре через символ перевода строки в алфавитном порядке или пустой ответ, если по данному числу невозможно получить никакого слова. После ответа, опишите как он был получен.

Problem 3: ПОЛИЗ

Напишите программу-калькулятор выражений в обратной польской записи над множествами.
Программе на стандартный вход подаются символьные строки, разделенные пробельными символами. Каждая символьная строка представляет собой либо запись множества, либо символ операции. Символьная строка — это последовательность непробельных символов. Пробельный символ — это символ пробела, или перевода строки, или табуляции.
Множество представляется строкой из цифр, заглавных и строчных латинских букв. Таким образом может кодироваться множество из 62 элементов. Нормализованной

записью множества является запись, в которой сначала следуют цифры, затем заглавные латинские буквы, затем строчные латинские буквы. Например, для множества Aa3Za его нормализованной записью будет 3AZa.
Пустое множество обозначается одиночным символом #.
Над множествами допустимы операции: & (пересечение), | (объединение), ^ (симметрическая разность) и ~ (дополнение до полного множества из 62 элементов, то есть результатом операции дополнения являются элементы множества, которые отсутствовали в исходном множестве).
Гарантируется, что в результате вычисления выражения на стеке останется единственный элемент. Он должен быть напечатан в нормализованном виде на стандартный поток вывода.
Максимальная глубина стека в процессе вычислений не превышает 1000.

Examples

Input

```
a
  x |
```

Output

```
ax
```

RFC1924_2

В этой задаче требуется преобразовать IPv6 адрес из компактной формы в соответствии с RFC1924 в классическое представление. IPv6 адрес предстваляет из себя 16-байтное число которое записывается в шестнадцатиричной системе счисления и группы из 2х байт разделяются двоеточием. При этом если две и более групп подряд равны 0000, то они могут быть опущены и заменены на двойное двоеточие. Незначащие старшие нули в группах могут быть опущены.
Таким образом длина записи IPv6 в класической форме занимает от 2 до 39 символов.
Примеры адресов:

```
FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
 1080::8:800:200C:417A
```

В стандарте RFC1924 можно записать адрес с помощью не более 20 символов. Адрес рассматривается как 128-битное целое число, которое записывается в системе счисления с основанием 85 с использованием следующих символов (в порядке возрастания значений):

```
'0'..'9', 'A'..'Z', 'a'..'z', '!', '#', '$', '%', '&', '(',
')', '*', '+', '-', ';', '<', '=', '>', '?', '@', '^', '_',
'`', '{', '|', '}', и '~'.
```

Например, адрес 1080:0:0:0:8:800:200C:417A это число 21932261930451111902915077091070067066 в десятичной системе счисления, или 4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51 в 85-ричной. Что дает запись

4)+k&C#VzJ4br>0wv%Yp
Напишите программу, которая преобразует адрес из компактной формы в классическую запись IPv6. В записи необходимо использовать заглавные буквы и адрес должен иметь минимальную длину.
Пример ввода
4)+k&C#VzJ4br>0wv%Yp
Пример вывода
1080::8:800:200C:417A

Problem 5: Пароль - 2

Ваша задача — взломать генератор случайных паролей, основанный на использовании псевдослучайных чисел.
Известно, что для генерации паролей используется линейный конгруентный генератор псевдослучайных чисел с параметрами A = 1103515245, C = 12345, M = 2^{31}. Символы случайного пароля получаются последовательным получением псевдослучайного числа и его преобразованием в вещественное число на полуинтервале [0;1), а затем в целое число на полуинтервале [0;62), которое преобразовывается в символ цифры, заглавной или строчной латинской буквы в том порядке, в котором они находятся в кодовой таблице ASCII. Преобразования псевдослучайного числа в вещественное и обратно в целые — равномерные.
Например, если дана затравка 128123812 и требуемая длина пароля — 16 символов, будет получен пароль kD5X7la4Z9LZvD6F. Первое псевдослучайное число будет 1616688397, которое по описанным выше правилам будет преобразовано в букву k и т. д..
На стандартный поток ввода вашей программе подается пароль, в котором часть символов неизвестна. Неизвестные символы обозначаются символом "точка". Длина пароля не превышает 32 символа. Гарантируется, что первый символ всегда известен. Программа должна восстановить пароль полностью и вывести его на стандартный поток вывода. Если пароль невозможно восстановить однозначно, либо такой пароль не может быть сгенерирован описанным выше способом, выведите один символ "решетка".

Examples

Input

a.....

Output

#

Input

aaaaa...

Output

#

Input

aaaab...

Output

aaaabezy

Problem 6: Список

Напишите тесты для следующей задачи.
Дано определение типа звена двусвязного списка строк:

```
struct Node
{
   struct Node *prev, *next; // "ссылки" на предыдущий и следующий элемент
   char *elem;               // строка, которая хранится в элементе списке
};
```

И определение типа списка

```
struct List
{
   struct Node *first, *last; // "ссылки" на начало и конец списка
};
```

Напишите функцию process, обрабатывающую список за один проход от первого до последнего элемента по нему следующим образом.
Функции на вход передается список и строка str
Звенья, у которых строка elem равна строке str удаляются, а звенья, у которых строка elem лексикографически больше строки str переставляются в конец списка. Прочие звенья не изменяются.
При удалении звена необходимо освобождать память, занимаемую звеном и строкой.
В списке нет заглавного звена и он не закольцован. Строка str не пустая.
Программа выполняет следующие действия:

- Считывает первую строку текста стандартного потока ввода и отбрасывает у считанной строки хвостовые пробельные символы. Считанная строка будет передана в функцию process в параметре str. Строка str не может быть пустой.
- Считывает последующие строки текста до признака конца файла, отбрасывает у считанных строк хвостовые пробельные символы и формирует из них список, который будет передан первым аргументом функции process. Порядок строк в списке совпадает с порядком строк в файле.
- Вызывает функцию process с указанным списом и строкой.
- Распечатывает на стандартный поток вывода элементы списка сначала в прямом, потом в обратном порядке, по одному элементу на строку вывода.
- Освобождает память, занимаемую списком.

Ваша задача — написать тесты для проверки решений этой задачи. Множество тестов должно быть корректным и полным. Корректность означает, что любое правильное

решение исходной задачи должно успешно проходить все тесты. Полнота означает, что любое разумно неправильное решение исходной задачи должно не проходить хотя бы один тест. Под разумно неправильным решением понимается решение, которое не выдает намеренно неправильный ответ в зависимости от случайного входного набора или для входных данных, не следующих из условия задачи. Например, "неразумное" неправильное решение может давать неправильный ответ только если строка равна "1231412421". Такие "неразумные" неправильные решения рассматривать не нужно.
Тесты состоят из набора тестовых пар <входные данные;правильный ответ>. Входные данные должны быть корректными и удовлетворять ограничениям задачи. Правильный ответ должен быть корректен.
Файлы с входными данными должны называться 001.dat, 002.dat, 003.dat и т. д. Соответствующие файлы с правильным ответом должны называться 001.ans, 002.ans, 003.ans и т. д. Размер любого файла не должен превышать 16KiB. Файлы могут содержать только байты с кодами 13 (\r), 10 (\n), 32-126.
Тестовый набор необходимо сдать на проверку в виде архива в формате .tgz. Архив должен содержать единственный каталог с именем tests, в котором должны размещаться разработанные вами тесты. Количество тестов не должно превышать 30.
Для создания архива формата .tgz в командной строке выполните команду:
tar cfz arch.tgz tests
Второй вариант создания архива с помощью 7zip в два шага - создать tar-архив и потом этот факт сжать с помощью gzip

Examples

Input

```
p
a
p
z
```

Output

```
a
z
z
a
```

Папайя

Откапывая исчезнувший город древней цивилизации Папайя, археологи обнаружили довольно странный механизм, который при ближайшем рассмотрении оказался вычислительной машиной. Исходные данные машина читает с медленно задвигаемого во входное отверстие бамбукового стебля, на котором могут быть записаны числа в виде десятичных дробей и латинские буквы (пробелы могут быть использованы для

разделения стоящих рядом чисел, в остальных случаях пробелы игнорируются). Чтобы показать, что ввод данных закончен, стебель в нужном месте отпиливают.

Результаты вычислений машина вырезает автоматическим резцом на другом бамбуковом стебле, причём числа с ненулевой дробной частью она почему-то всегда отображает в виде десятичных дробей с шестью знаками после запятой; если же число целое, то дробная часть для него не отображается. Центральный процессор машины оснащён двадцатью шестью регистрами, которые обозначаются заглавными латинскими буквами от A до Z. К каждому регистру прилагается ещё некое подобие оперативного запоминающего устройства, организованного в виде стека, причём ячеек в каждом из 26 стеков оказалось довольно много - во всяком случае, исследователи быстро сбились со счёта. Каждый регистр, а также каждая ячейка каждого стека могут в любой момент времени содержать в качестве значения число (целое или дробное), символ (при этом в алфавит машины входят заглавные и строчные латинские буквы, точка, запятая, вопросительный и восклицательный знаки, а также пробел), либо специальное "пустое" значение, обозначаемое []. Значение "пусто" также используется машиной для обозначения логической лжи, любые другие значения считаются истинными.

Поскольку регистр A участвует во всех вычислениях, археологи решили называть его аккумулятором; кроме того, любой из 26 регистров, в том числе аккумулятор, может быть назначен текущим, причём в начале работы текущим считается как раз аккумулятор. При выполнении любой арифметической операции машина берёт первый операнд из аккумулятора, второй операнд - из текущего регистра, результат помещает обратно в аккумулятор.

Программа для машины Папайя представляет собой строку символов, каждый из которых задаёт машинную команду; программа, состоящая из символов-команд, выполняется последовательно слева направо, кроме двух команд, которые могут нарушить эту последовательность. Команды a, b, c и все остальные строчныелатинские буквы означают "занести данную букву в аккумулятор", а команда :(двоеточие) означает "изменить регистр буквы в аккумуляторе на противоположный", при этом заглавная буква меняется на строчную, строчная на заглавную, точка меняется на восклицательный знак и обратно, запятая меняется на вопросительный знак и обратно, если же в аккумуляторе что-то другое, то ничего не происходит. Команды A, B, C и все остальные заглавные латинские буквы означают "назначить данный регистр текущим". Команды 0, 1, ..., 9 означают "занести в аккумулятор соответственно число 0.0, 1.0, ... 9.0". Команда @ означает "занести в аккумулятор пробел". Команды +, -, *, / означают соответствующие арифметические действия, ^ означает возведение в степень, команды <, >, = означают сравнение двух чисел или двух символов, & и | означают логическое "и" и логическое "или"; все эти команды используют значение из аккумулятора в качестве левого операнда, значение из текущего регистра в качестве правого операнда, результат заносится обратно в аккумулятор. Команда # умножает аккумулятор на десять, команда _ делит аккумулятор на десять. Команда !. работает как логическое отрицание аккумулятора: если там значение "пусто", то заносится значение 1, если любое другое - заносится значение "пусто".

Команда $ выдаёт текущее значение аккумулятора на печать, при этом буквы, знаки препинания и пробелы печатаются как они есть, числа с нулевой дробной частью печатаются как целые, остальные числа печатаются с пятью знаками после запятой; команда ? вводит очередное число или букву в аккумулятор, а если входной бабмуковый стебель кончился, заносит в аккумулятор значение "пусто". Следует обратить внимание, что на входном стебле любая последовательность цифр, возможно, разделённая одной точкой, и, возможно, с минусом в начале, воспринимается как единое значение (число) точка воспринимается как обычный символ только в случае, если она идёт не после цифры, либо если в данной последовательности цифр это уже вторая точка.

Команда] заносит значение из текущего регистра в его стек (напомним, что стек у каждого регистра свой); команда [извлекает значение с вершины стека текущего регистра и заносит его в текущий регистр (если извлекать нечего, в регистр заносится значение "пусто"); команда ~ меняет местами значения в текущем регистре и на вершине его стека, а если стек пуст, заносит в регистр значение "пусто", а в стек - бывшее содержимое регистра.

Команда) копирует значение из аккумулятора в текущий регистр, команда (копирует значение из текущего регистра в аккумулятор.

Команды { и } предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс фигурных скобок. Выполняются они так. Команда {, если в аккумуляторе "пусто", идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в аккумуляторе было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда }, наоборот, если в аккумуляторе "пусто", не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной фигурной скобки, после чего продолжает выполнение с команды, стоящей после такой скобки справа.

Наконец, команда " прекращает выполнение программы, при этом выполнение считается успешным. Если программа кончилась, не встретив эту команду, она завершается аварийно. Пробелы в программе игнорируются
Список команд

- a-z - занести букву; 0-9 - занести число; @ -- занести пробел
- : - сменить регистр символа; A-Z - сменить текущий регистр
- +, -, *, /, - арифм. операция; ^ - возведение в степень
- <, >, = - операция сравнения; &, | - лог.операция
- # - умножение на 10; _ - деление на 10
- ! - отрицание аккумулятора
- ? - ввод; $ - вывод
- [- из стека текущего регистра;] - в стек текущего регистра
- ~ - поменять местами текущий регистр и вершину его стека
-) - из аккумулятора в текущий регистр; (- из текущего регистра в аккумулятор

- { - если ложь, то вперёд до парной скобки; } - если истина, то назад до парной скобки
- '' - стоп (успех)

Требуется написать для него программу, которая попоследовательности, которая состоит из различных натуральных чисел и завершается числом 0 определияет значение второго по величине элемента в этой последовательности.
Числа, следующие за числом 0, считывать не нужно.

Examples

Input

1 7 9 0

Output

7

Найти квадрат

Мальчик Казимир нарисовал квадрат, но так как в комнате было темно, он его потерял. Помогите мальчику Казимира найти квадрат. Казимир может спросить про какую-то точку и узнать минимальное расстояние от нее до сторон квадрата.
Напишите программу, которая с помощью таких вопросов узнает координаты вершин квадрата.
После каждого вывода координаты ('W' и два числа в одной строке) на стандартный поток вывова программа может считать со стандартного потока одно вещественное число - расстояние до квадрата.
Как только квадрат будет найден - выведите координаты квадрата - 'F' и 4 пары вещественных чисел и завершите работу программы.
После всех выводов программа должна очистить буфер с помощью flush.
Координаты квадрата целочисленные, в запросе координаты не могут превышать по модулю 10^6. Число запросов не должно превышать 10000.

Examples

Input

W -1 0
W -1 1
W 2 0
W 2 1
W 0 -1
W 1 -1
W 0 2
W 1 2

F 0 0 0 1 1 1 1 0

Output

1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0