Отборочный-1 10-11

Распаковка числа

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10]
Распакуйте строку 1[11]0[1]0[1[11]]10[10[11]1]1[1011]0 в запись двоичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в троичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в троичной системе и закрывающей скобки –]. Так число 11111111000 в упакованной записи будет записано следующим образом: 1[2[2]]0[10]
Распакуйте строку 1[11]2[2[2]]1[1]0[12]1[1[10]]0[1] в запись троичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10]
Распакуйте строку 1[1010]10[10]01[10]10[10[1[11]]]10[101] в запись двоичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в троичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в троичной системе и закрывающей скобки –]. Так число 11111111000 в упакованной записи будет записано следующим образом: 1[2[2]]0[10]
Распакуйте строку 1[1[10]]0[2]1[10]2[2[1]]10[11] в запись троичного числа.

Последовательность

Последовательность чисел 2303, 511, 3311, 1519, 2527, 735, 1743, 2751, 959, 1967, ... , 1792 была получена следующим образом. Сначала были выписаны 500 первых натуральных чисел, кратных 7, по возрастанию (7, 14, … 3500). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (007, 00E, …, DAC). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (700, E00, …, CAD). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FF8, FF1, …, 007). Затем строки были обратно преобразованы в исходные числа (2303, 511, …, 1792). Определите номер места, на котором стоит в последовательности число 3024 (места нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 4095, 1791, 2799, 495, 3807, 1503, 2511, 207, 3519, 1215, ... , 2304 была получена следующим образом. Сначала были выписаны 455 первых натуральных чисел, кратных 9, по возрастанию (9, 18, … 4095). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (009, 012, …, FFF). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (900, 210, …, FFF). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FFF, FF6, …, 009). Затем строки были обратно преобразованы в исходные числа (4095, 1791, …, 2304). Определите число, которое стоит в последовательности на 239-м месте (нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 2559, 1791, 1023, 255, 2799, 2031, 1263, 495, 2271, 1503, ... , 768 была получена следующим образом. Сначала были выписаны 1000 первых натуральных чисел, кратных 3, по возрастанию (3, 6, … 3000). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (003, 006, …, BB8). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (300, 600, …, 8BB). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FF9, FF6, …, 003). Затем строки были обратно преобразованы в исходные числа (2559, 1791, …, 768). Определите номер места, на котором стоит в последовательности число 1971 (места нумеруются с 1).

Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 4095, 2815, 1535, 255, 3055, 1775, 495, 3295, 2015, 735, ... , 1280 была получена следующим образом. Сначала были выписаны 819 первых натуральных чисел, кратных 5, по возрастанию (5, 10, … 4095). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (005, 00A, …, FFF). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (500, A00, …, FFF). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FFF, FFA, …, 005). Затем строки были обратно преобразованы в исходные числа (4095, 2815, …, 1280). Определите число, которое стоит в последовательности на 745-м месте (нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Формула

Написать программу, которая по заданному целому положительному числу до 10000 получает минимальную по длине формулу дающую это число. Формула может состоять из символов '2', '3', '+', '*'.
Если получить число невозможно, выведите "No solution" без кавычек

Examples

Input

5

Output

3+2

Input

11

Output

3*3+2

Распаковка числа - 2

Рассмотрим способ упакованной записи чисел в p-чной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в p-чной системе и закрывающей скобки –]. Так число 111111111000 в упакованной записи в двоичной системе может быть записано следующим образом: 1[10[10]1]0[11]
Составьте программу, которая распаковывает число в p-чной системе счисления.

Input format

В первой строке вводится основание системы счисления p (1 < p < 11). Во второй строке находится упакованное число.

Output format

Выведите распакованное число. Гарантируется, что длина распакованного числа не превосходит 100000 символов

Examples

Input

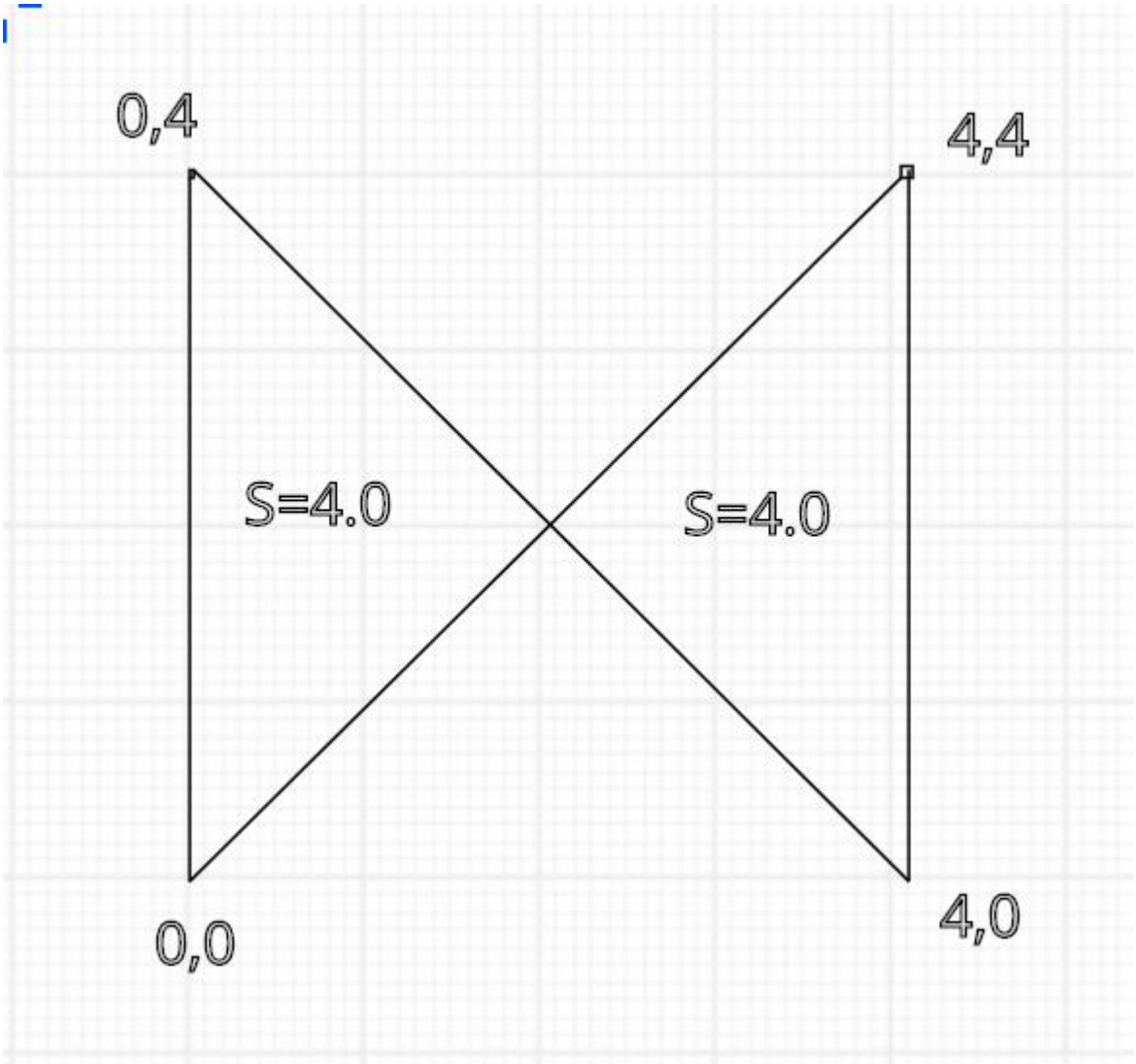
2
1[1010]000

Output

1111111111000

Площади

Задана замкнутая ломаная, напишите программу, которая находит конечные площади каждой области, которые эта ломаная образует.



Input format

В первой строке вводится N - число точек замкнутой ломаной (2 < N < 11). В следующих N строках целочисленные координаты точек ломаной на плоскости.

Output format

Выведите ненулевые площади конечных областей в порядке возрастания с точностью 1e-6

Examples

Input

```
4
0 0
0 4
4 0
4 4
```

Output

4.000000
4.000000

Problem 6: Пересечение отрезков

Дана задача. На стандартном потоке ввода подаются четыре 32-битных знаковых целых чисел L1, H1, L2, H2. Числа L1, H1 задают отрезок на числовой прямой вида [L1;H1) (то есть L1 входит в отрезок, а H1 — не входит). Гарантируется, что L1 <= H1, если L1 == H1, такой отрезок задает пустое множество. Числа L2, H2 задают границы второго отрезка на числовой прямой аналогично первому отрезку.
На стандартный поток вывода напечатайте пару 32-битных знаковых целых чисел L3, H3, задающих отрезок на числовой прямой, который является пересечением отрезков [L1;H1) и [L2;H2). Если результатом пересечения является пустое множество, оно всегда выводится в виде "0 0".
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 значение H2 равно 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. В данном случае, входные данные должны представлять собой четыре целых числа, представимых 32-битным знаковым типом, с указанными ограничениями на границы. Числа могут разделяться пробельными символами произвольным образом.
Правильный ответ должен быть корректен. В данном случае, правильный ответ должен представлять собой два числа, являющихся правильным ответом на входные данные. Числа могут разделяться пробельными символами произвольным образом.
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 1KiB.
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 превышаться 20.
Для создания архива формата .tgz в командной строке выполните команду:
tar cfz arch.tgz tests
Или воспользуйтесь архиватором 7zip

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 справа налево, 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)

В вашем распоряжении оказался эмулятор данного компьютера) и требуется написать для него программу, которая по заданному на ленте году выводит строку "YES", если он високосный и "NO" в противном случае (без кавычек). Год является високосным в двух случаях: либо он кратен 4, но при этом не кратен 100, либо кратен 400. Год не является високосным, если он не кратен 4, либо он кратен 100, но при этом не кратен 400.

Examples

Input

2014

Output

NO

Input

2004

Output

YES

Площадь суши

В некотором плоском клетчатом прямоугольном мире (N на M клеток) существует 2 континента. Континент представляет из себя связанный набор клеток. Континенты не имеют общих точек. Черные клетки (обозначаются 1) – суша, белые клетки (обозначаются 0) – вода.
На одном из них живет красный квадрокоптер, который мечтает съездить в путешествие на другой континент. Квадрокоптер умеет двигаться вверх, вниз, вправо, влево (соответствующие команды U, D, R, L), хранить в памяти любой кусочек карты и видеть в радиусе одной клетки (в том числе по диагонали). После перемещения на стандартный поток ввода будет записано то, что видит квадрокоптер вокруг себя. У квадрокоптера зарядки хватает на 8 действий движения. После истощения зарядки он приземляется на клетку, над которой находится. Если это суша – он может сесть и

зарядиться от солнечной энергии. Если это вода – квадрокоптер тонет. Принудительная посадка квадрокоптера – P. Утверждается, что между 2 континентами существует путь, занимающий не более 5 действий квадрокоптера. Ваша задача определить площадь суши. Как только программа готова вывести ответ напечатайте W и одно число - площадь суши. В начале работы программы на стандартном потоке ввода находится информация о том, что видит квадрокоптер вокруг себя. После каждого вывода требуется произвести операцию очистки буфера вывода. Примеры кода для этого действия можно найти на http://ejudge.cs.msu.ru/kvadrocopter

Input format

В начале работы программы и после каждого перемещения ввядится 9 чисел 0 или 1 - квадрат, который видит квадрокоптер

Output format

Выводится одна из команд перемещения или W и число, когда ответ найден

Examples

Input

```
0 0 0
0 1 0
0 0 0

0 0 0
1 0 1
0 0 0

0 0 0
0 1 0
0 0 0
```

Output

```
R
R
W 2
```

Отборочный-1 5-9

Распаковка числа

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной

системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10]
Распакуйте строку 1[11]0[1]0[1[11]]10[10[11]1]1[1011]0 в запись двоичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в троичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в троичной системе и закрывающей скобки –]. Так число 11111111000 в упакованной записи будет записано следующим образом: 1[2[2]]0[10]
Распакуйте строку 1[11]2[2[2]]1[1]0[12]1[1[10]]0[1] в запись троичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в двоичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 1111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в двоичной системе и закрывающей скобки –]. Так число 111111100 в упакованной записи будет записано следующим образом: 1[1[1[10]]]0[10]
Распакуйте строку 1[1010]10[10]01[10]10[10[1[11]]]10[101] в запись двоичного числа.

Распаковка числа

Рассмотрим способ упакованной записи чисел в троичной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в троичной системе и закрывающей скобки –]. Так число 11111111000 в упакованной записи будет записано следующим образом: 1[2[2]]0[10]
Распакуйте строку 1[1[10]]0[2]1[10]2[2[1]]10[11] в запись троичного числа.

Последовательность

Последовательность чисел 2303, 511, 3311, 1519, 2527, 735, 1743, 2751, 959, 1967, ... , 1792 была получена следующим образом. Сначала были выписаны 500 первых натуральных чисел, кратных 7, по возрастанию (7, 14, … 3500). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (007, 00E, …, DAC). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (700, E00, …, CAD). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FF8, FF1, …, 007). Затем строки были обратно преобразованы в

исходные числа (2303, 511, …, 1792). Определите номер места, на котором стоит в последовательности число 3024 (места нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 4095, 1791, 2799, 495, 3807, 1503, 2511, 207, 3519, 1215, ... , 2304 была получена следующим образом. Сначала были выписаны 455 первых натуральных чисел, кратных 9, по возрастанию (9, 18, … 4095). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (009, 012, …, FFF). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (900, 210, …, FFF). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FFF, FF6, …, 009). Затем строки были обратно преобразованы в исходные числа (4095, 1791, …, 2304). Определите число, которое стоит в последовательности на 239-м месте (нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 2559, 1791, 1023, 255, 2799, 2031, 1263, 495, 2271, 1503, ... , 768 была получена следующим образом. Сначала были выписаны 1000 первых натуральных чисел, кратных 3, по возрастанию (3, 6, … 3000). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (003, 006, …, BB8). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (300, 600, …, 8BB). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FF9, FF6, …, 003). Затем строки были обратно преобразованы в исходные числа (2559, 1791, …, 768). Определите номер места, на котором стоит в последовательности число 1971 (места нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Последовательность

Последовательность чисел 4095, 2815, 1535, 255, 3055, 1775, 495, 3295, 2015, 735, ... , 1280 была получена следующим образом. Сначала были выписаны 819 первых натуральных чисел, кратных 5, по возрастанию (5, 10, … 4095). Затем все числа были переведены в шестнадцатеричную систему, причём каждое число было записано тремя цифрами (005, 00A, …, FFF). Затем каждая запись числа была преобразована в строку и символы в этой строке были записаны в обратном порядке (500, A00, …, FFF). После чего полученные строки были отсортированы в порядке, обратном алфавитному (FFF, FFA, …, 005). Затем строки были обратно преобразованы в исходные числа (4095, 2815, …, 1280). Определите число, которое стоит в последовательности на 745-м месте (нумеруются с 1).
Указание: Для решения можно использовать электронные таблицы.
При сдаче задачи в первой строке запишите ответ, в следующих строках приведите описание процесса получения ответа (это может быть - описание действий, программа и т.д.)

Формула

Написать программу, которая по заданному целому положительному числу до 10000 получает минимальную по длине формулу дающую это число. Формула может состоять из символов '2', '3', '+', '*'.
Если получить число невозможно, выведите "No solution" без кавычек

Examples

Input

5

Output

3+2

Input

11

Output

3*3+2

Распаковка числа - 2

Рассмотрим способ упакованной записи чисел в p-чной системе. Идея способа состоит в том, чтобы одинаковые подряд идущие цифры, например 111111111, заменять на конструкцию, состоящую из повторяемой цифры (в примере – 1), открывающей скобки перед числом повторений – [, упакованной записи числа повторений в p-чной системе и закрывающей скобки –]. Так число 111111111000 в упакованной записи в двоичной системе может быть записано следующим образом: 1[10[10]1]0[11]
Составьте программу, которая распаковывает число в p-чной системе счисления.

Input format

В первой строке вводится основание системы счисления p (1 < p < 11). Во второй строке находится упакованное число.

Output format

Выведите распакованное число. Гарантируется, что длина распакованного числа не превосходит 100000 символов

Examples

Input

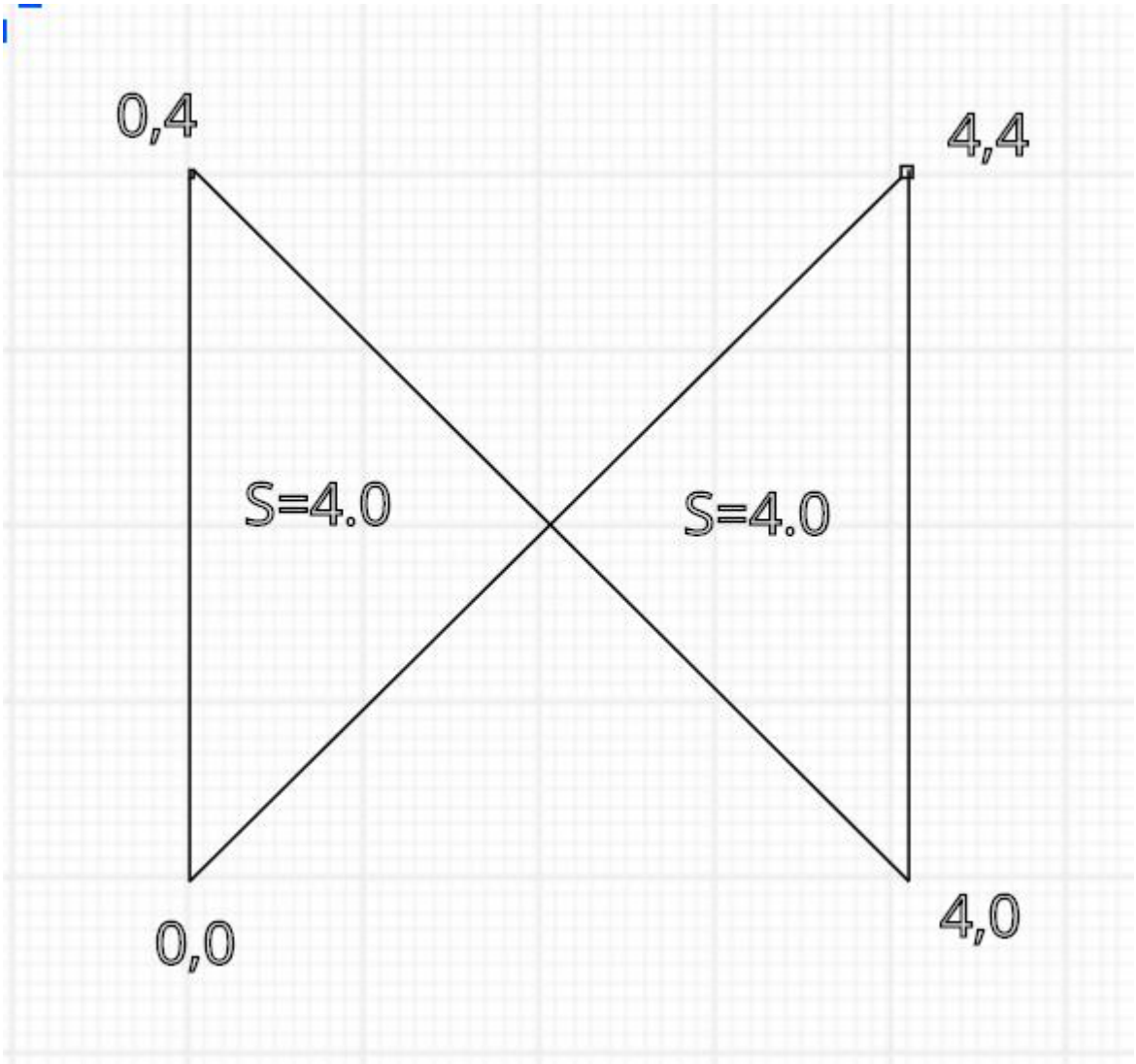
2
1[1010]000

Output

1111111111000

Площади

Задана замкнутая ломаная, напишите программу, которая находит конечные площади каждой области, которые эта ломаная образует.



Input format

В первой строке вводится N - число точек замкнутой ломаной (2 < N < 11). В следующих N строках целочисленные координаты точек ломаной на плоскости.

Output format

Выведите ненулевые площади конечных областей в порядке возрастания с точностью 1e-6

Examples

Input

```
4
0 0
0 4
4 0
4 4
```

Output

4.000000
4.000000

Problem 6: Пересечение отрезков

Дана задача. На стандартном потоке ввода подаются четыре 32-битных знаковых целых чисел L1, H1, L2, H2. Числа L1, H1 задают отрезок на числовой прямой вида [L1;H1) (то есть L1 входит в отрезок, а H1 — не входит). Гарантируется, что L1 <= H1, если L1 == H1, такой отрезок задает пустое множество. Числа L2, H2 задают границы второго отрезка на числовой прямой аналогично первому отрезку.
На стандартный поток вывода напечатайте пару 32-битных знаковых целых чисел L3, H3, задающих отрезок на числовой прямой, который является пересечением отрезков [L1;H1) и [L2;H2). Если результатом пересечения является пустое множество, оно всегда выводится в виде "0 0".
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 значение H2 равно 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. В данном случае, входные данные должны представлять собой четыре целых числа, представимых 32-битным знаковым типом, с указанными ограничениями на границы. Числа могут разделяться пробельными символами произвольным образом.
Правильный ответ должен быть корректен. В данном случае, правильный ответ должен представлять собой два числа, являющихся правильным ответом на входные данные. Числа могут разделяться пробельными символами произвольным образом.
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 1KiB.
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 превышаться 20.
Для создания архива формата .tgz в командной строке выполните команду:
tar cfz arch.tgz tests
Или воспользуйтесь архиватором 7zip

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 справа налево, 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)

В вашем распоряжении оказался эмулятор данного компьютера) и требуется написать для него программу, которая по заданному на ленте году выводит строку "YES", если он високосный и "NO" в противном случае (без кавычек). Год является високосным в двух случаях: либо он кратен 4, но при этом не кратен 100, либо кратен 400. Год не является високосным, если он не кратен 4, либо он кратен 100, но при этом не кратен 400.

Examples

Input

2014

Output

NO

Input

2004

Output

YES

Площадь суши

В некотором плоском клетчатом прямоугольном мире (N на M клеток) существует 2 континента. Континент представляет из себя связанный набор клеток. Континенты не имеют общих точек. Черные клетки (обозначаются 1) – суша, белые клетки (обозначаются 0) – вода.
На одном из них живет красный квадрокоптер, который мечтает съездить в путешествие на другой континент. Квадрокоптер умеет двигаться вверх, вниз, вправо, влево (соответствующие команды U, D, R, L), хранить в памяти любой кусочек карты и видеть в радиусе одной клетки (в том числе по диагонали). После перемещения на стандартный поток ввода будет записано то, что видит квадрокоптер вокруг себя. У квадрокоптера зарядки хватает на 8 действий движения. После истощения зарядки он приземляется на клетку, над которой находится. Если это суша – он может сесть и

зарядиться от солнечной энергии. Если это вода – квадрокоптер тонет. Принудительная посадка квадрокоптера – P. Утверждается, что между 2 континентами существует путь, занимающий не более 5 действий квадрокоптера. Ваша задача определить площадь суши. Как только программа готова вывести ответ напечатайте W и одно число - площадь суши. В начале работы программы на стандартном потоке ввода находится информация о том, что видит квадрокоптер вокруг себя. После каждого вывода требуется произвести операцию очистки буфера вывода. Примеры кода для этого действия можно найти на http://ejudge.cs.msu.ru/kvadrocopter

Input format

В начале работы программы и после каждого перемещения ввядится 9 чисел 0 или 1 - квадрат, который видит квадрокоптер

Output format

Выводится одна из команд перемещения или W и число, когда ответ найден

Examples

Input

```
0 0 0
0 1 0
0 0 0

0 0 0
1 0 1
0 0 0

0 0 0
0 1 0
0 0 0
```

Output

```
R
R
W 2
```

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

Отборочный-2 5-9

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
```

Ответы на отборочный этап (10-11 классы)

1) Распаковка числа - 1

1) 1110000000100000000000000000111111111110
2) 11112222222210000011111111111110
3) 1111111111100011100000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000010
0000
4) 1111111111111001112210000

2) Последовательность

1) 474
2) 3735
3) 767
4) 1905

3) Требовалось написать программу, используя метод динамического программирования или/и предподсчитать ответ для каждого числа.

4) Требовалось написать программу реализующую алгоритм распаковки.

5) Требовалось написать программу, общий ход вычислений предлагается следующий:

1) Найти все точки пересечения ломаной между собой
2) Для каждой тройки точек - проверить лежит ли он целиком внутри замкнутой фигуры, образующей ломаной
3) Для каждого такого треугольника определить какие его стороны принадлежат ломаной
4) Объединить треугольники по общим сторонам не принадлежащим ломаным

6) Пример тестов

0 1 1 2
1 2 0 1
-100400 -100200 600000 700000
600000 700000 -100400 -100200
-1000000 -1000000 10 100000
-1000000000 1000000000 -5 -5
100000 100000 -1000000000 1000000000
-1000000000 1000000000 -100 100
500 501 -1000000000 1000000000
1000 2000 1000 1999
1000 1999 1000 2000
1000 2000 1001 2000
1001 2000 1000 2000
-111111 123123 7784 666666

7784 666666 -111111 123123
Пример ответов к ним
0 0
0 0
0 0
0 0
0 0
0 0
0 0
-100 100
500 501
1000 1999
1000 1999
1001 2000
1001 2000
7784 123123
7784 123123

7) Пример решения

?D)
4##B)C)
D<A{C(B+C)D<A}
C(D=A{y:$e:$s:$"}
1##B)C)
D<A{C(B+C)D<A}
C(D=A{n:$o:$"}
4B)C)
D<A{C(B+C)D<A}
C(D=A{y:$e:$s:$"}
n:$o:$"

8) Общий ход решения:

1) Обходом в глубину построить карту первого континента, определить его площадь
2) С учетом запаса топлива по границам континента найти соседний
3) Вторым обходом в глубину найти его площадь

Ответы на отборочный этап (5-9 классы)

1) Распаковка числа - 1

   1) 11100000000100000000000000000111111111110
   2) 11112222222210000011111111111110
   3) 1111111111100011100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100000
   4) 1111111111111001112210000

2) Последовательность

   1) 474
   2) 3735
   3) 767
   4) 1905

3) Требовалось написать программу, используя метод динамического программирования или/и предподсчитать ответ для каждого числа.

4) Требовалось написать программу реализующую алгоритм распаковки.

5) Требовалось написать программу, общий ход вычислений предлагается следующий:

   1) Найти все точки пересечения ломаной между собой
   2) Для каждой тройки точек - проверить лежит ли он целиком внутри замкнутой фигуры, образующей ломаной
   3) Для каждого такого треугольника определить какие его стороны принадлежат ломаной
   4) Объединить треугольники по общим сторонам не принадлежащим ломаным

6) Пример тестов

```
0 1 1 2
1 2 0 1
-100400 -100200 600000 700000
600000 700000 -100400 -100200
-1000000 -1000000 10 100000
-1000000000 1000000000 -5 -5
100000 100000 -1000000000 1000000000
-1000000000 1000000000 -100 100
500 501 -1000000000 1000000000
1000 2000 1000 1999
1000 1999 1000 2000
1000 2000 1001 2000
1001 2000 1000 2000
-111111 123123 7784 666666
```

7784 666666 -111111 123123
Пример ответов к ним
0 0
0 0
0 0
0 0
0 0
0 0
0 0
-100 100
500 501
1000 1999
1000 1999
1001 2000
1001 2000
7784 123123
7784 123123

7) Пример решения

?D)
4##B)C)
D<A{C(B+C)D<A}
C(D=A{y:$e:$s:$"}
1##B)C)
D<A{C(B+C)D<A}
C(D=A{n:$o:$"}
4B)C)
D<A{C(B+C)D<A}
C(D=A{y:$e:$s:$"}
n:$o:$"

8) Общий ход решения:

1) Обходом в глубину построить карту первого континента, определить его площадь
2) С учетом запаса топлива по границам континента найти соседний
3) Вторым обходом в глубину найти его площадь

Замечание

Во всех задачах, если не указано особое, программа должна производить ввод со стандартного потока ввода или файл input.txt, выводить на стандартный поток или файл output.txt, ограниеение по времени 2 с, ограничение по памяти 256MB.

Задача A. Цифровой корень

Цифровой корень натурального числа — сумма цифр от суммы цифр от суммы цифр ... числа. Например, для числа 345 цифровой корень 3 (345 -> 12 -> 3)

Ваша задача для заданного цифрового корня K найти N-значное число, цифровой корень которого равен K и в котором нет подряд двух одинаковых цифр

Формат входных данных

Вводится два целых числа N и K ($1 \leqslant N \leqslant 10^5, 0 \leqslant K \leqslant 9$)

Формат выходных данных

Требуется вывести искомое число без ведущих нулей или -1, если такого числа не существует

Пример

стандартный ввод	стандартный вывод
3 3	345

Задача B. Перфекционист Глеб

Холодным мартовским утром Глеб играл на улице в игру: для начала он придумывает перестановку из чисел от 1 до N — a, также запоминая a_R - перестановку получающаяся из a разворотом. Потом он выбирает число K, и выписывает K раз подряд последовательность, полученную склеиванием a и a_R. Глеб очень любит чтобы у полученной последовательности длина наибольшей возрастающей подпоследовательности равнялась N. Помогите Глеб найти такое минимальное K.

Формат входных данных

Первая строка входных данных содержит число N ($1 \leqslant N \leqslant 300\,000$)

Во второй строке записаны N целых чисел a_i ($1 \leqslant a_i \leqslant n$, $a_i \neq a_j$ для $i \neq j$) — загаданная перестановка.

Формат выходных данных

В единственной строке выведите искомое K. Гарантируется, что такое K существует.

Примеры

стандартный ввод	стандартный вывод
3 3 2 1	1
3 2 3 1	2
4 3 4 1 2	2
4 4 3 1 2	1

Задача C. Связность графа

Однажды на день рождение Ивану подарили граф состоящий из N вершин в котором вершины с номерами x и y соединены ребром тогда и только тогда, когда $gcd(x, y) \geqslant g$, где

Формат входных данных

Первая строка входных данных содержит три целых числа N, g, M и ($1 \leqslant N, g, M \leqslant 200\,000$) — количество вершин в графе и параметр графа и количество запросов на связность.

Далее следуют m строк, описывающих запросы, каждый запрос задан двумя числами x и y, ($1 \leqslant x, y \leqslant N, x \neq y$).

Формат выходных данных

Выведите M, по одной на каждый запрос, соответственно «Yes» , если вершины находятся в одной компоненте, и «No» иначе.

Пример

стандартный ввод	стандартный вывод
6 2 4 2 3 4 3 5 1 2 6	Yes Yes No Yes

Задача D. ЧПУ-фрезер

В этой задаче вам требуется написать программу для фрезерного станка с ЧПУ по изображению заготовки. Изображение задается в виде прямоугольника $N \cdot M$,

где N – количество строк, а M – количество столбцов. Символ '.' означает, что эта клетка не должна быть обработана. Символ '*' означает, что эта клетка должна быть отфрезерована.

Фреза станка изначально находиться в точке с координатой $(0, 0)$ (левый верхний угол картинки) и может за одну секунду переместиться в одну из соседних клеток (команды `N` - вверх, `E` - вправо, `S` - вниз , `W` - влево). Кроме того, в каждый момент времени фреза может находиться или в опущенном состоянии и тогда происходит процесс обработки, или в поднятом состоянии и тогда обработка не происходит. Подъем и опускание фрезы занимает 1 секунду. Обработка клетки происходит за одну секунду.

В процессе обработки заготовки происходит нагрев фрезы в опущенном состоянии на 1 градус за одну секунду и охлаждение на 1 градус в поднятом состоянии. Соответственно, команда D позволяет опустить фрезу и команда U позволяет поднять фрезу. Будем считать, что время опускания и поднятия фрезы пренебрежимо малы.

Начальная и минимальная температура 30 градусов. При нагреве больше 50 градусов фреза ломается. Если фреза выходит за границы изображения, то она ломается.

Ваша задача сгенерировать программу для фрезера из команд `N, E, S, W, U, D`, которая получает заданное изображение.

Программа для фрезера оценивается по времени получения изображения. Программы которые приводят к поломке фрезы считаются неверными.

Формат входных данных

Изображение описывается числами N и M ($1 \leqslant N, M \leqslant 1000$), затем идет N строк по M символов '.' или '*'

Формат выходных данных

Вывести требуется программу из команд 'N','E','S','W','U','D' в одну строку без пробела, которую должен выполнить станок для получения изображения. Длина программы фрезера не должна превышать 10^6 символов.

Система оценки

Итоговый балл за каждый тест и подзадачу будет выставляться в зависимости от лучшего полученного результата среди участников после олимпиады.

Пример

стандартный ввод	стандартный вывод
5 5 ....* ....* *...* ....* *****	EEEEDSUWWWWSDUEEEEDSUWWWWSDEEEE

Замечание

В задаче есть несколько подзадач с файлам изображений, для которых требуется сгенерировать и отправить на проверку файл с программой. Отдельной подзадачей требуется отправить программу на одном из языков программирования, которая для произвольного изображения получает программу для станка.

При проверке файлов с изображениями будет доступен протокол проверки корректности программы для станка и время работы для программы не приводящей к поломке.

Подзадача со сдачей программы на языке программирования будет проверяться на всех тестовых изображениях во время олимпиады и на некотором наборе тестов после окончания олимпиады.

Задача E. RW

Многопоточность — свойство платформы (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

В этой задаче требуется написать программу, которая корректно работает многопоточно. Есть область общей памяти, позволяющая чтение и запись. Несколько параллельно работающих функций имеют к ней доступ, при этом одновременно могут читать сколько угодно потоков, но писать — только один. Необходимо обеспечить корректный доступ к этой памяти.

В этой задаче требуется написать программу на псевдоязыке для функции читателей области памяти и писателей в эту область. Так как параллельное чтение и запись в общую область памяти может привести к некорректному поведения, требуется избежать случая, когда несколько пишущих функций могут одновременно писать и случая когда читатель читает, а писатель пишет одновременно.

Реализации такого ограничения могут быть различными, но нас интересует случай приоритета писателей. Как только появился хоть один писатель, читателей

больше не пускать. При этом читатели могут простаивать.

Для того чтобы обеспечить логику ожидания существует специальная общая переменная-*mutex* для которой есть две операции *enter* и *leave*. Операция *enter* успешно выполняется если никакая другая функция не выполнила с этой переменной операцию *enter* не выполнив операцию *leave*. В противном случае выполнения функции останавливается до тех пор, пока та функция, которая сделала операцию *enter* не выполнит операцию *leave*. Если на операции *enter* ожидает несколько функций, то успешно выполнить её может только одна.

Вы можете завести не более 10 глобальных переменных-*mutex* и не более 10 целочисленных переменных изначально равных нулю:

```
global
  int x;
  int y;
  mutex z;
```

где `int x` - объявление целочисленной переменной `x`, `mutex z` - объявление переменной-*mutex* `x`. Имя переменной может состоять из одной латинской строчной буквы. Имена переменных не могу повторяться.

Прототип функции читателя выглядит так:

```
reader_begin
//  тут можно добавить свой код вместо комментария
read()
// тут можно добавить свой код вместо комментария
reader_end
```

Прототип функции писателя выглядит так:

```
writer_begin
// тут можно добавить свой код вместо комментария
write()
// тут можно добавить свой код вместо комментария
writer_end
```

В качестве операций можно использовать следующие конструкции:

- `z.enter();` - операция `enter` для переменной-`mutex` z
- `z.exit();`
- `x.increase();` - увеличить целочисленную переменную `x` на 1
- `x.decrease();` - уменьшить целочисленную переменную `x` на 1
- `if (x==<константа>) <другая операция>` выполнение операции при выполнении условия

При выполнении read() и write() и происходят соответствующие чтения и записи общей памяти.

Формат выходных данных

Необходимо сдать программу на этом псевдоязыке, которая решает задачу. Например,

```
global
  int x;
  mutex y;

reader_begin
  x.increase();
  read();
  if (x == 1) x.decrease();
reader_end

writer_begin
  y.enter();
  write();
  y.leave();
writer_end
```

В программе должны остаться вызовы read() и write()

Замечание

Программа из примера не является правильным решением и приведена только для иллюстрации формата. Считайте что функции читателей и писателей могут работать параллельно в нескольких экземплярах

Задача F. Квадрокоптеры

Однажды Федор решил провести чемпионат мира по программированию. Ясно что такое соревнование нельзя провести без торжественного открытия, а открытие без красочного шоу. К счастью у Федора есть N квадрокоптеров и он решил построить из них заданную фигуру в зале над участниками.

Управление квадрокоптера осуществляется следующим образом. Изначально в квадрокоптер загружается программа, которая каждый момент времени решает куда должен переместиться квадрокоптер - влево, вправо, вверх, вниз или остаться на месте (эти команды кодируются числами от 1 до 5 соответственно). Каждый момент времени каждый квадрокоптер принимает решение о перемещении. После этого все квадрокоптеры одномоментно совершают это перемещение.

У каждого квадрокоптера есть 10 байт памяти которую он может изменять каждый момент времени. У каждого квадрокоптера есть сенсоры которые позволяют считать состояние памяти соседних (по сторонам) квадрокоптеров или квадрокоптеров в той же клетке.

Представим зал в виде прямоугольной сетки, к некоторых клетках которой находятся квадрокоптеры. Изначально они находятся в неизвестных различных координатах, но гарантируется что начальная конфигурация является связной по сторонам фигурой.

Вам требуется написать программу которая выполняется на квадрокоптере.

При формировании фигуры квадрокоптерами важно только относительное положение квадрокоптеров относительно друг-друга. То есть требуется чтобы в какой-то момент времени относительная конфигурация должна совпадать с заданной.

Формат входных данных

Каждый момент времени на вход программе подается неизменяемая целевая конфигурация квадрокоптеров: число N ($1 \leqslant N \leqslant 20$) и N пар целых положительных чисел (каждое из которых до 100) - точки в которых могут находиться квадрокоптеры, чтобы сформировать нужную конфигурацию.

Следующие 10 чисел задают состояние памяти текущего квадрокоптера, на котором выполняется программа - числа от 0 до 255, разделенные проблемами

Затем следует число соседних квадрокоптеров M. Следующие M строк задают состояние каждого из квадрокоптеров - первое число от 1 до 5 где он находится (слева, справа, сверху, снизу, в той же клетке) и 10 чисел от 0 до 255 - состояние его памяти.

Формат выходных данных

Требуется вывести текущее действие квадрокоптера - число от 1 до 5, затем 10 байт - новое состояние памяти

Замечание

Для каждого хода для каждого квадрокоптера запускается решение с данными, описанными во входном формате. Программа делающая один ход не должна предполагать что можно как-то сохранить данные кроме как в 10 байтах памяти квадрокоптера.

Пример ввода для одного хода:

```
4
0 0
1 1
2 2
3 3
0 1 200 0 0 0 0 0 0 0
3
1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0 0
```

Пример вывода:

```
2
0 1 199 0 0 0 0 0 0 0
```

Замечание

Во всех задачах, если не указано особое, программа должна производить ввод со стандартного потока ввода или файл input.txt, выводить на стандартный поток или файл output.txt, ограниеение по времени 2 с, ограничение по памяти 256MB.

Задача A. Цифровой корень

Цифровой корень натурального числа — сумма цифр от суммы цифр от суммы цифр ... числа. Например, для числа 345 цифровой корень 3 (345 -> 12 -> 3)

Ваша задача для заданного цифрового корня K найти N-значное число, цифровой корень которого равен K.

Формат входных данных

Вводится два целых числа N и K ($1 \leqslant N \leqslant 10^5, 1 \leqslant K \leqslant 9$)

Формат выходных данных

Требуется вывести искомое число без ведущих нулей.

Пример

стандартный ввод	стандартный вывод
3 3	345

Задача B. Перфекционист Глеб

Холодным мартовским утром Глеб играл на улице в игру: для начала он придумывает перестановку из чисел от 1 до N — a, также запоминая a_R - перестановку получающаяся из a разворотом. Потом он выбирает число K, и выписывает K раз подряд последовательность, полученную склеиванием a и a_R. Глеб очень любит чтобы у полученной последовательности длина наибольшей возрастающей подпоследовательности равнялась N. Помогите Глеб найти такое минимальное K.

Формат входных данных

Первая строка входных данных содержит число N ($1 \leqslant N \leqslant 300\,000$)

Во второй строке записаны N целых чисел a_i ($1 \leqslant a_i \leqslant n$, $a_i \neq a_j$ для $i \neq j$) — загаданная перестановка.

Формат выходных данных

В единственной строке выведите искомое K. Гарантируется, что такое K существует.

Примеры

стандартный ввод	стандартный вывод
3 3 2 1	1
3 2 3 1	2
4 3 4 1 2	2
4 4 3 1 2	1

Задача C. Связность графа

Однажды на день рождение Ивану подарили граф состоящий из N вершин в котором вершины с номерами x и y соединены ребром тогда и только тогда, когда $gcd(x, y) \geqslant g$, где

Формат входных данных

Первая строка входных данных содержит три целых числа N, g, M и ($1 \leqslant N, g, M \leqslant 200\,000$) — количество вершин в графе и параметр графа и количество запросов на связность.

Далее следуют m строк, описывающих запросы, каждый запрос задан двумя числами x и y, ($1 \leqslant x, y \leqslant N, x \neq y$).

Формат выходных данных

Выведите M, по одной на каждый запрос, соответственно «Yes» , если вершины находятся в одной компоненте, и «No» иначе.

Пример

стандартный ввод	стандартный вывод
6 2 4 2 3 4 3 5 1 2 6	Yes Yes No Yes

Задача D. ЧПУ-фрезер

В этой задаче вам требуется написать программу для фрезерного станка с ЧПУ по изображению заготовки. Изображение задается в виде прямоугольника $N \cdot M$,

где N – количество строк, а M – количество столбцов. Символ '.' означает, что эта клетка не должна быть обработана. Символ '*' означает, что эта клетка должна быть отфрезерована.

Фреза станка изначально находиться в точке с координатой $(0, 0)$ (левый верхний угол картинки) и может за одну секунду переместиться в одну из соседних клеток (команды `N` - вверх, `E` - вправо, `S` - вниз , `W` - влево). Кроме того, в каждый момент времени фреза может находиться или в опущенном состоянии и тогда происходит процесс обработки, или в поднятом состоянии и тогда обработка не происходит. Подъем и опускание фрезы занимает 1 секунду. Обработка клетки происходит за одну секунду.

В процессе обработки заготовки происходит нагрев фрезы в опущенном состоянии на 1 градус за одну секунду и охлаждение на 1 градус в поднятом состоянии. Соответственно, команда D позволяет опустить фрезу и команда U позволяет поднять фрезу. Будем считать, что время опускания и поднятия фрезы пренебрежимо малы.

Начальная и минимальная температура 30 градусов. При нагреве больше 50 градусов фреза ломается. Если фреза выходит за границы изображения, то она ломается.

Ваша задача сгенерировать программу для фрезера из команд `N, E, S, W, U, D`, которая получает заданное изображение.

Программа для фрезера оценивается по времени получения изображения. Программы которые приводят к поломке фрезы считаются неверными.

Формат входных данных

Изображение описывается числами N и M ($1 \leqslant N, M \leqslant 1000$), затем идет N строк по M символов '.' или '*'

Формат выходных данных

Вывести требуется программу из команд 'N','E','S','W','U','D' в одну строку без пробела, которую должен выполнить станок для получения изображения. Длина программы фрезера не должна превышать 10^6 символов.

Система оценки

Итоговый балл за каждый тест и подзадачу будет выставляться в зависимости от лучшего полученного результата среди участников после олимпиады.

Пример

стандартный ввод	стандартный вывод
5 5 ....* ....* *...* ....* *****	EEEEDSUWWWWSDUEEEEDSUWWWWSDEEEE

Замечание

В задаче есть несколько подзадач с файлам изображений, для которых требуется сгенерировать и отправить на проверку файл с программой. Отдельной подзадачей требуется отправить программу на одном из языков программирования, которая для произвольного изображения получает программу для станка.

При проверке файлов с изображениями будет доступен протокол проверки корректности программы для станка и время работы для программы не приводящей к поломке.

Подзадача со сдачей программы на языке программирования будет проверяться на всех тестовых изображениях во время олимпиады и на некотором наборе тестов после окончания олимпиады.

Задача E. Квадрокоптеры

Однажды Федор решил провести чемпионат мира по программированию. Ясно что такое соревнование нельзя провести без торжественного открытия, а открытие без красочного шоу. К счастью у Федора есть N квадрокоптеров и он решил построить из них заданную фигуру в зале над участниками.

Управление квадрокоптера осуществляется следующим образом. Изначально в квадрокоптер загружается программа, которая каждый момент времени решает куда должен переместиться квадрокоптер - влево, вправо, вверх, вниз или остаться на месте (эти команды кодируются числами от 1 до 5 соответственно). Каждый момент времени каждый квадрокоптер принимает решение о перемещении. После этого все квадрокоптеры одномоментно совершают это перемещение.

У каждого квадрокоптера есть 10 байт памяти которую он может изменять каждый момент времени. У каждого квадрокоптера есть сенсоры которые позволяют считать состояние памяти соседних (по сторонам) квадрокоптеров или квадрокоптеров в той же клетке.

Представим зал в виде прямоугольной сетки, к некоторых клетках которой находятся квадрокоптеры. Изначально они находятся в неизвестных различных ко-

ординатах, но гарантируется что начальная конфигурация является связной по сторонам фигурой.

Вам требуется написать программу которая выполняется на квадрокоптере.

При формировании фигуры квадрокоптерами важно только относительное положение квадрокоптеров относительно друг-друга. То есть требуется чтобы в какой-то момент времени относительная конфигурация должна совпадать с заданной.

Формат входных данных

Каждый момент времени на вход программе подается неизменяемая целевая конфигурация квадрокоптеров: число N ($1 \leqslant N \leqslant 20$) и N пар целых положительных чисел (каждое из которых до 100) - точки в которых могут находиться квадрокоптеры, чтобы сформировать нужную конфигурацию.

Следующие 10 чисел задают состояние памяти текущего квадрокоптера, на котором выполняется программа - числа от 0 до 255, разделенные проблемами

Затем следует число соседних квадрокоптеров M. Следующие M строк задают состояние каждого из квадрокоптеров - первое число от 1 до 5 где он находится (слева, справа, сверху, снизу, в той же клетке) и 10 чисел от 0 до 255 - состояние его памяти.

Формат выходных данных

Требуется вывести текущее действие квадрокоптера - число от 1 до 5, затем 10 байт - новое состояние памяти

Замечание

Для каждого хода для каждого квадрокоптера запускается решение с данными, описанными во входном формате. Программа делающая один ход не должна предполагать что можно как-то сохранить данные кроме как в 10 байтах памяти квадрокоптера.

Пример ввода для одного хода:

```
4
0 0
1 1
2 2
3 3
0 1 200 0 0 0 0 0 0 0
3
1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0 0
```

Пример вывода:

```
2
0 1 199 0 0 0 0 0 0 0
```

Ответы и критерии проверки заключительного этапа.

Тесты, проверяющие программы, и ответы находятся в архиве тестирующей системы:
https://ejudge.cs.msu.ru/lomo18.tar.gz
Протоколы проверки и технические баллы доступны по логину и паролю участника на сайте
https://ejudge.cs.msu.ru/ej/client?contest_id=104&locale_id=1
https://ejudge.cs.msu.ru/ej/client?contest_id=105&locale_id=1

Критерии проверки (10-11 классы):
1 - 101 тест, по 1 баллу за тест, кроме первого, за него 0.
2 - 72 теста, по 1 баллу за первые 44 теста, за остальные 2 балла.
3 - 55 тестов, по 1 баллу за первые 10 тестов, остальные по 2.
4 - D1-D5 50 баллов, по 10 баллов за каждую в зависимости от полученного ответа, критерии на число баллов в checker_1.cpp - checker_5.cpp. D - 50 баллов - сумма по каждому тесту, отмасштабированная согласно valuer.py.
5 - 20 тестов, с разбалловкой test_score_list = "2 2 2 2 2 4 4 4 4 4 2 2 2 2 2 4 4 4 4 4", максимальный балл 100 за полное решение.
6 - 10 тестов по 10 баллов.

В 5-9 классе аналогично без 5-й задачи.

Оценка выставлялась по следующей схеме: в 10-11 классах балл, полученный из критериев делился на 431, затем умножался на 100 и округлялся до ближайшего целого, а в 5-9 классах балл, полученный из критериев делился на 350, затем умножался на 100 и округлялся до ближайшего целого.