Замечание

Во всех задачах программа должна производить ввод со стандартного потока ввода или файл input.txt. Выводить на стандартный поток или файл output.txt. Во всех задача ограниияение по времени 2 с. Ограничение по памяти 256MB.

Задача A. RFC3092

В соответсвии со «стандартом»(RFC3092) именования, переменные и функции в программе называются `foo, bar, baz, qux, quux, quuux, quuuux, ...`. Имена переменных выбираются по номеру их первого упоминания в тексте фрагмента программы.

Вашей задачей является переименование переменных в примере кода, так чтобы они соответствовали RFC3092.

Формат входных данных

На вход подается фрагмент кода, каждая из строк которого является выражением присваивания. Выражение присваивания формально описывается следующим образом

⟨*выражение присванивания*⟩ ::= ⟨*имя переменной*⟩ '=' ⟨*выражение*⟩

⟨*имя переменной*⟩ ::= строка из строчных латинских букв

⟨*выражение*⟩ ::= ⟨*число*⟩ | ⟨*имя переменной*⟩ | ⟨*выражение*⟩ ⟨*знак операции*⟩ ⟨*выражение*⟩

⟨*число*⟩ ::= строка из цифор

⟨*знак операции*⟩ ::= '+' | '-' | '/' | '*'

Операнды и операции разделяются ровно одним пробелом. Общий размер входных данных не превышает 1000000 символов

Формат выходных данных

Выведите текст программы после переименования переменных в том же формате. Переменные с одинаковым именем во входной программе должны называться одинаково в выходной. Переменные должны называться в соответсвии с RFC3092 в порядке их первого вхождения в фрагмент кода.

Примеры

standard input	standard output
a = 1 b = a + b	foo = 1 bar = foo + bar

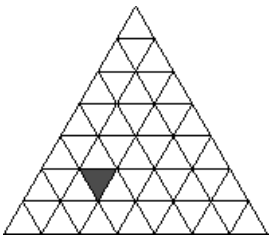
Замечание

Эта задача состоит из 3 частей — в первых двух частях требуется отправить результат после преобразования двух файлов, которые доступны в тестирующей системе. В третьей части требуется отправить на проверку программу, которая производит переименование. Стоимости подзадач 20%/30%/50%.

Задача B. Треугольная игра

Двое играют на треугольной доске (см. рис.), закрашивая по очереди на ней треугольные клеточки. Одна клетка (начальная) уже закрашена перед началом игры.

Первым ходом закрашивается клеточка, граничащая (по стороне) с начальной, а каждым следующим ходом — клетка, граничащая с только что закрашенной. Повторно клетки красить нельзя. Тот, кто не может сделать ход, проигрывает.



Напишите программу, которая для текущего начального положения находит ход, который приведет к выигрышу игрока.

Формат входных данных

В первой строке задается t - число начальных положений в игре, для каждого из которых требуется найти ответ.

Для каждого положения игры вводится число N ($1 < N < 10$) – размер треугольного поля. Затем число от 1 до N^2, номер начальной покрашенной клетки. Клетки поля нумеруются с единицы слева-направо в строке, по строкам сверху-вниз.

Формат выходных данных

Для каждого теста в отдельной строке выведите номер клетки, куда нужно пойти, чтобы выиграть при оптимальной стратегии второго игрока. Если выиграть невозможно - выведите ноль.

Примеры

standard input	standard output
2 2 3 2 1	0 3

Замечание

Эта задача состоит из 2 частей — в первой части требуется отправить результат для файла, которые доступного в тестирующей системе. Во второй части требуется отправить на проверку программу, которая производит вычисление. Стоимости подзадач 30%/70%.

Задача C. Диагонали в клетках

Вася взял клетчатый лист бумаги и нарисовал там прямоугольник $N \times M$. Затем в каких-то клеточках он провел диагонали таким образом чтобы никакие две диагонали не имели общих точек. Найдите число различных рисунков, которые Вася мог нарисовать.

Формат входных данных

На стандартном потоке задано два числа N и M $(0 < N < 5, 0 < M < 20)$.

Формат выходных данных

Выведите единственное число — число различных рисунков по модулю 1000000007.

Примеры

standard input	standard output
1 2	7

Замечание

Эта задача состоит из 2 частей — в первой части требуется отправить результат для файла, которые доступного в тестирующей системе. Во второй части требуется отправить на проверку программу, которая производит вычисление. Стоимости подзадач 30%/70%

Задача D. Шифр простой замены

Шифр простой замены — класс методов шифрования, которые сводятся к созданию по определённому алгоритму таблицы шифрования, в которой для каждой буквы открытого текста существует единственная сопоставленная ей буква шифр-текста. Само шифрование заключается в замене букв согласно таблице. Для расшифровки достаточно иметь ту же таблицу, либо знать алгоритм, по которой она генерируется. Для вскрытия подобных шифров используется частотный криптоанализ. (Википедия)

Напишите программу, которая будет расшифровывать текст на английском языке, зашифрованный шифром простой замены. Для сбора статистической информации об английских текстах на вход программе будет подаваться корпус незашифрованных текстов. Кроме того известно, что все слова в зашифрованном тексте встречаются в корпусе текстов.

Формат входных данных

На стандартном потоке ввода вашей программе сначала будет подаваться корпус обучающих текстов, затем зашифрованный текст. И в обучающих текстах, и в зашифрованном тексте слова состоят только из заглавных и строчных латинских букв. Слова отделяются друг от друга произвольным количеством пробельных символов (то есть пробелов, табуляций или символов конца строки). Регистр букв в словах не значим. Обучающие тексты заканчиваются символом '-' (минус), который находится в отдельной строке. Входной текст имеет концы строк Unix (один символ с кодом 10), но для программ на PascalABC входной текст имеет концы строк DOS (два символа с кодами 13 10).

Формат выходных данных

На стандартный поток вывода напечатайте расшифрованный текст. Если какую-либо букву зашифрованного текста невозможно расшифровать однозначно, вместо нее выводите символ '?' (знак вопроса). Слова в расшифрованном тексте разделяйте одним знаком пробела, но формируйте строки выходного файла так, чтобы длина каждой строки не превышала 64 символа (не считая символов конца строки). Если длина одного слова превышает 64, то выводите одно такое слово в строке, не разрезая его на части. В расшифрованном тексте сохраняйте регистр букв исходного текста.

Примеры

Пример входных данных и результата будет на странице сдачи задания в тестирующую систему.

Задача E. Машина 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 **строчные** латинские 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)

Задача состоит из 4-х подзадача, первые три смотрите в тестирующей системе.

Подзадача 4

Напишите для машины Папайя программу, которая читает последовательность чисел произвольной длины (то есть читает числа, пока они не кончатся) и выдаёт наибольшее из введённых чисел, наименьшее из введённых чисел и среднее арифметическое всех введённых чисел.

Замечание

После отправки на проверку ответов на первые две части вам будет доступна сдача четвертой части. Для четвертой части в тестирующей системе будет доступна ссылка на исполняемый файл, эмулирующий работу машины Папайя.

Эмулятор запускается содним параметром, который должен представлять собой имя файла с программойдля машины Папайя; в качестве входного стебля используется поток стандартного ввода, результаты эмулятор печатает в поток стандартного вывода. Стоимости подзадач 10

Задача F. Пересечение звездчатых многоугольников

Дан многоугольник с вершинами в целочисленных координатах. Будем считать, что точки A и B, лежащие внутри многоугольника, **видны** друг из друга, если отрезок AB лежит внутри многоугольника.

Назовем **ядром** многоугольника множестов точек из которых видны все точки многоугольника. Назовем многоугольник **звёздчатым**, если его ядро не пусто.

Требуется написать программу, которая по двум заданным звездчатым многоугольникам находит число областей их пересечения.

Формат входных данных

На стандартном потоке в первой задано число вершин первого звездного многоугольника ($2 < N < 20$). Затем в N идут N пар целых чисел x_i, y_i – координаты вершин первого многоугольника в порядке обхода против часовой стрелки. Затем идет аналогичное описание второго многоугольника.

Все координаты не превышают по модулю 100.

Формат выходных данных

На стандартный поток вывода выведити единственное число – число областей пересечения двух многоугольников ненулевой площади.

Примеры

standard input	standard output
4 0 0 10 3 1 1 3 10 4 10 10 0 3 9 9 3 0	2

Замечание

Эта задача состоит из 2 частей — в первой части требуется отправить результат для файла, которые доступного в тестирующей системе. Во второй части требуется отправить на проверку программу, которая производит вычисление.

Стоимости подзадач 30%/70%