Олимпиада "Ломоносов" по информатике. 2016-17. 10-11 класс. Первый отборочный тур.

Задача 1.1: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 1$,
$a_2 = 1$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 114 по 187 символ (символы нумеруются с единицы).

Задача 1.2: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 0$,
$a_2 = 1$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 124 по 178 символ (символы нумеруются с единицы).

Задача 1.3: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 1$,
$a_2 = 0$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 102 по 165 символ (символы нумеруются с единицы).

Задача 1.4: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 1$,
$a_2 = 1$,
$a_3 = 0$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 110 по 196 символ (символы нумеруются с единицы).

Задача 2: Запись числа

На вход программе подаётся положительное целое число N (N < 1000001) и последовательность из N символов. Требуется определить, можно ли из введённой последовательности выбрать символы (в любом порядке) так, чтобы получилась запись шестнадцатеричного числа по правилам языка Си. В языке Си запись шестнадцатеричного числа начинается с нуля (0), за ним следует строчная буква икс (x), за которой следуют одна или более чем одна шестнадцатеричная цифра (любая из: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Составлять запись числа можно только из тех “буквенных” цифр, которые подходят по регистру, то есть, являются заглавными. Любая цифра в записи числа может быть использована не большее количество раз, чем количество её вхождений во введенную последовательность. Если можно составить запись хотя бы для одного числа, то нужно вывести запись для наибольшего из возможных чисел. Если запись составить нельзя, программа выводит No

Примеры

Входные данные
17
I-AM-A-PROGRAMMER

Выходные данные
No

Входные данные
17
1xAMxAxPR0GRaMMER

Выходные данные
0xEAA1

Задача 3.1: Художник

Для создания новой картины мальчик Казимир взял клечатый лист бумаги размера 3*6. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

```
01
11

10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.2: Художник

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 4*5. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

```
01
11

10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.3: Художник

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 6*3. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.

Например на поле 2x2 могло получиться 4 картины

```
01
11

10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.4: Художник

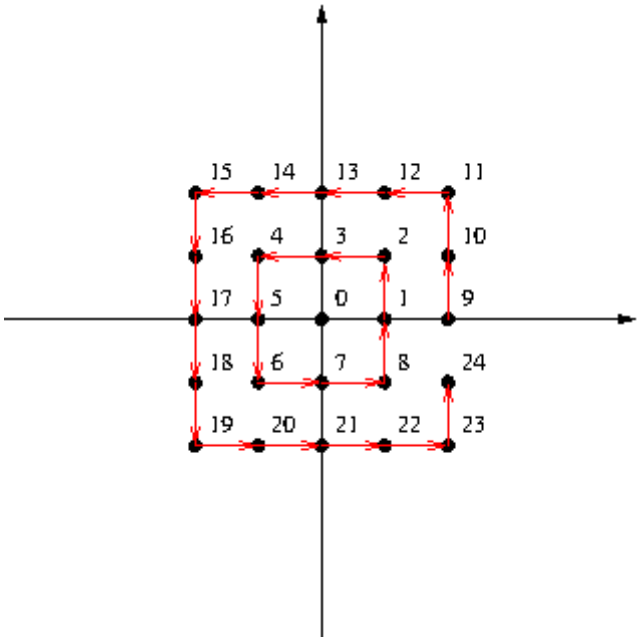
Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 5*4. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

```
01
11

10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 4: Прямые и точки

Рассмотрим координатную плоскость. Занумеруем точки с целочисленными координатами следующим образом. Точка (0,0) получает номер 0. Точка (1,0) получает номер 1, (1,1) получает номер 2, (0,1) получает номер 3 и так далее против часовой стрелки.



Напишите программу, определяющую принадлежность точек прямой. Сначала на стандартном потоке ввода задаются два номера точек, определяющие прямую на плоскости. Точки не совпадают. Затем задается последовательность номеров точек, принадлежность которых заданной прямой требуется проверить. Последовательность заканчивается числом -1.
Для каждой точки в последовательности на стандартный поток вывода напечатайте 0, если эта точка не находится на заданной прямой, и 1, если точка находится на прямой.
Все номера точек - 32-битные положительные целые числа.
Число проверяемых точек во входном потоке не превышает 500000.

Примеры

Входные данные
16 2 12 3 6 -1

Выходные данные
0 1 0

Задача 5: Сжатие изображения

Важный раздел информатики - это алгоритмы сжатия изображений. В этой задаче мы будем рассматривать только сжатие черно-белых картинок без потерь. Задача помехоустойчивого кодирования заключается в том, что входные бинарные данные преобразовываются в закодированные бинарные данные таким образом, чтобы в размер сжатых данных был минимальным. Разработайте алгоритм и напишите программу, которая реализовывает наиболее эффективное по вашему мнению алгоритм сжатия и распаковки черно-белых изображений.

На вход программе подается либо изображение для сжатия, либо поток данных для распаковки, а именно: на стандартном потоке ввода (т. е., например, с клавиатуры) программе сначала задается число 0, если программа должна сжать данные, либо число 1, если программа должна распаковать данные.
В случае кодирования далее на вход подается картинка 51 строк по 100 символов. Цифра ‘0’ соответствует белому цвету, цифра ‘1’ соответствует черному. Все прочие символы (пробелы, переводы строк) должны игнорироваться.
В случае сжатия данных программа должна вывести на стандартный поток вывода (экран) закодированные данные как строка из '0' и '1' и завершить работу
В случае декодирования данных программа должна вывести на стандартный поток вывода (экран) изображение из 51 строк по 100 символов '0' или '1'.
Программа в режиме распаковки данных должна распаковывать данные, сжатой той же самой вашей программой в режиме закодирования данных. Никаких других данных для распаковки подаваться не будет.
Ваша задача - придумать и реализовать такой алгоритм сжвтия и распаковки, который имел бы наилучшую эффективность, то есть размер сжатой картинки. Тестирование будет производиться на наборе тестовом и контрольных наборах. Баллы за каждое изображение будут выставляться в зависимости от степени сжатия
Пример входных данных для кодирования (в условии размер картинки для простоты 3x3):

```
0
010
000
000
```

Возможный пример результата работы:

```
0001
```

Пример входных данных для декодирования:

```
1
0001
```

Результат работы:

```
010
000
000
```

Задача 6: Художник-2

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 4*N. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна

покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется оценить число вариантов картин, которые могли получиться.
На стандартном потоке вводится число N < 10001
Выведите одно число - количество различных картин, которые могли получиться по модулю 1000000007

Примеры

Входные данные
2

Выходные данные
10

Задача 7: Анти-Бармаглот

Компания *Barmaley's Computing* учла пожелания пользователей и выпустила вторую версию компьютера - *Barmaglot-2*. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер остался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека могут содержать либо число, либо латинскую букву или пробел, либо специальное значение [BARMALEY], которое используется для обозначения логической лжи, при том что любое другое обозначает истину.

Программа для Barmaglot'а 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 латинские буквы означают "занести данную букву в регистр".
- Команды 0, 1, ..., 9 означают "занести в регистр соответствующее число".
- Команда @ означает занести в регистр пробел.

- Команды +, -, *, /, % означают соответствующие арифметические действия, операции целочисленные. % - операция взятия остатка
- Команды <, >, = означают сравнение двух чисел или двух символов, & и | означают логическое "и" и логическое "или";

все эти команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.

- Команда # умножает содержимое регистра в десять раз,
- Команда _ делит содержимое регистра в десять раз.
- Команда ! работает как логическое отрицание содержимого регистра: если там значение [BARMALEY], то заносится значение 1, если любое другое - заносится значение [BARMALEY].
- Команда . выдаёт текущее значение регистра на печать
- команда ? вводит очередное число(целиком, а не по разрядам), а если на вводе кончилась (оборвалась) лента, заносит в регистра значение [BARMALEY].
- Команда] заносит значение из регистра в стек;
- Команда [извлекает значение с вершины стека и заносит его в регистра (если извлекать нечего, в регистр заносится [BARMALEY]);
- Команда ~ меняет местами значения в регистра и на вершине стека.
- Команда } заносит значение из регистра в очередь,
- Команда { извлекает из очереди самое старое значение и помещает в аккумулятор (или помещает туда [BARMALEY], если очередь пуста).
- Команды (и) предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Команда (, если в регистре [BARMALEY], идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в регистре было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда), наоборот, если в регистре [BARMALEY], не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной круглой 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.

Пример программы, которая печатает традиционную строчку “HELLO WORLD”:
H.E.L.L.O.@.W.O.R.L.D."

Вам требуется написать программу которая эмулирует программу на этом языке программирования.

В первой строке вводится программа для Barmaglot. Во второй строке исходное содержимое ленты.

Требуется вывести результат работы программы или "[ERROR]" если выполнение завершилось с ошибкой или программа синтаксические неверная.

Пример работы эмулятора можно найти по ссылке
http://ejudge.cs.msu.ru/barmaglot2/

Примеры

Входные данные
]?(}}?)}!{{(]]{(~]{>([}}!])[(~}}!)[]]{()}[.@.{{)"
2 10 3 9

Выходные данные
2 3 9 10

Входные данные
H.H.
text

Выходные данные
[ERROR]

Задача 8: Звездчатый многоугольник

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

Требуется написать программу, которая по заданному звездчатому многоугольнику найти его ядро. Можно заметить что ядро звездчатого многоугольника тоже многоугольник

На стандартном потоке ввода задано число вершин звездного многоугольника ($2 < N < 20$). Затем идут N пар целых чисел x_i, y_i - координаты вершин многоугольника в порядке обхода против часовой стрелки. Все координаты не превышают по модулю 10000.
На стандартный поток вывода выведите число вершин ядра исходного многоугольника, затем вершины ядра против часовой стрелки в том же формате, что и во вводе. Среди всех возможных ответом выведите тот, в котором наименьшее число вершин.

Примеры

Входные данные

```
6
0 0
2 0
2 1
1 1
1 2
0 2
```

Выходные данные

```
4
0 0
1 0
1 1
0 1
```

Олимпиада "Ломоносов" по информатике. 2016-17. 10-11 класс. Второй отборочный тур.

Задача 1.1: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = (a_{n-1} + a_{n-2}) \bmod 1000$,
$a_1 = 1$,
$a_2 = 1$,
Где mod - операция взятия остатка от деления.
Все числа последовательности выписали в строку без разделителей.
Отправьте на проверку число с 114 по 187 символ включительно (символы нумеруются с единицы).

Задача 1.2: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = (a_{n-1} + a_{n-2}) \bmod 1000$,
$a_1 = 1$,
$a_2 = 1$,
Где mod - операция взятия остатка от деления.
Все числа последовательности выписали в строку без разделителей.
Отправьте на проверку число с 100 по 157 символ включительно (символы нумеруются с единицы).

Задача 1.3: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = (a_{n-1} + a_{n-2}) \bmod 1000$,
$a_1 = 1$,
$a_2 = 1$,
Где mod - операция взятия остатка от деления.
Все числа последовательности выписали в строку без разделителей.
Отправьте на проверку число с 124 по 197 символ включительно (символы нумеруются с единицы).

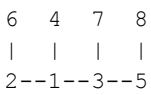
Задача 1.4: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = (a_{n-1} + a_{n-2}) \bmod 1000$,
$a_1 = 1$,
$a_2 = 1$,
Где mod - операция взятия остатка от деления.
Все числа последовательности выписали в строку без разделителей.
Отправьте на проверку число с 304 по 407 символ включительно (символы нумеруются с единицы).

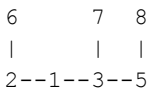
Задача 2.1: Код Прюфера

Кодирование Прюфера переводит помеченные деревья порядка n в последовательность чисел от 1 до n по алгоритму: Пока количество вершин больше двух:
1) Выбирается лист v с минимальным номером.
2) В код Прюфера добавляется номер вершины, смежной с v.

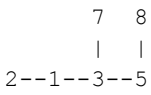
3) Вершина v и инцидентное ей ребро удаляются из дерева.
Полученная последовательность называется кодом Прюфер для заданного дерева.
Рассмотрим построение кода Прюфера на примере. Возьмем дерево:



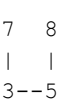
Наименьший лист в дереве 4, номер смежной вершины 1, поэтому в код Прюфера добавляем (1).



На следующем шаге Наименьший лист в дереве 6, номер смежной вершины 2, поэтому в код Прюфера добавляем 2 и получаем (1,2).



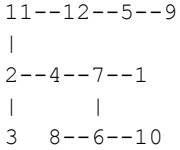
(1,2,1)



(1,2,1,3)

(1,2,1,3,3)

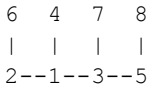
(1,2,1,3,3,5)
(Осталось 2 вершины, завершаем алгоритм. Получаем для примера код Прюфера (1,2,1,3,3,5).
Постройте код Прюфера для дерева:



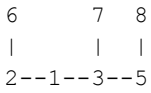
Ответ запишите в таком же виде, как в примере (то есть последовательность чисел записывается в скобках, числа разделяются запятой, пробелов в ответе быть не должно).

Задача 2.2: Код Прюфера

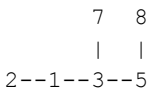
Кодирование Прюфера переводит помеченные деревья порядка n в последовательность чисел от 1 до n по алгоритму: Пока количество вершин больше двух:
1) Выбирается лист v с минимальным номером.
2) В код Прюфера добавляется номер вершины, смежной с v.
3) Вершина v и инцидентное ей ребро удаляются из дерева.
Полученная последовательность называется кодом Прюфер для заданного дерева.
Рассмотрим построение кода Прюфера на примере. Возьмем дерево:



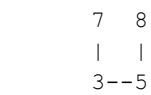
Наименьший лист в дереве 4, номер смежной вершины 1, поэтому в код Прюфера добавляем (1).



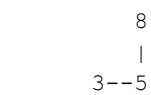
На следующем шаге Наименьший лист в дереве 6, номер смежной вершины 2, поэтому в код Прюфера добавляем 2 и получаем (1,2).



(1,2,1)

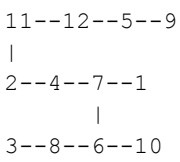


(1,2,1,3)



(1,2,1,3,3)

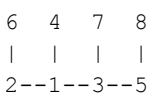
(1,2,1,3,3,5)
(Осталось 2 вершины, завершаем алгоритм. Получаем для примера код Прюфера (1,2,1,3,3,5).
Постройте код Прюфера для дерева:



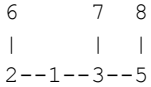
Ответ запишите в таком же виде, как в примере (то есть последовательность чисел записывается в скобках, числа разделяются запятой, пробелов в ответе быть не должно).

Задача 2.3: Код Прюфера

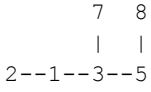
Кодирование Прюфера переводит помеченные деревья порядка n в последовательность чисел от 1 до n по алгоритму: Пока количество вершин больше двух:
1) Выбирается лист v с минимальным номером.
2) В код Прюфера добавляется номер вершины, смежной с v.
3) Вершина v и инцидентное ей ребро удаляются из дерева.
Полученная последовательность называется кодом Прюфер для заданного дерева.
Рассмотрим построение кода Прюфера на примере. Возьмем дерево:



Наименьший лист в дереве 4, номер смежной вершины 1, поэтому в код Прюфера добавляем (1).



На следующем шаге Наименьший лист в дереве 6, номер смежной вершины 2, поэтому в код Прюфера добавляем 2 и получаем (1,2).

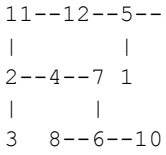


(1,2,1)

(1,2,1,3)

(1,2,1,3,3)

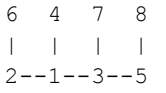
(1,2,1,3,3,5)
(Осталось 2 вершины, завершаем алгоритм. Получаем для примера код Прюфера (1,2,1,3,3,5).
Постройте код Прюфера для дерева:



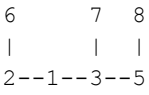
Ответ запишите в таком же виде, как в примере (то есть последовательность чисел записывается в скобках, числа разделяются запятой, пробелов в ответе быть не должно).

Задача 2.4: Код Прюфера

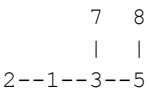
Кодирование Прюфера переводит помеченные деревья порядка n в последовательность чисел от 1 до n по алгоритму: Пока количество вершин больше двух:
1) Выбирается лист v с минимальным номером.
2) В код Прюфера добавляется номер вершины, смежной с v.
3) Вершина v и инцидентное ей ребро удаляются из дерева.
Полученная последовательность называется кодом Прюфер для заданного дерева.
Рассмотрим построение кода Прюфера на примере. Возьмем дерево:



Наименьший лист в дереве 4, номер смежной вершины 1, поэтому в код Прюфера добавляем (1).



На следующем шаге Наименьший лист в дереве 6, номер смежной вершины 2, поэтому в код Прюфера добавляем 2 и получаем (1,2).

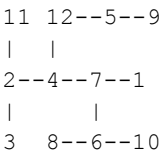


(1,2,1)

(1,2,1,3)

(1,2,1,3,3)

(1,2,1,3,3,5)
(Осталось 2 вершины, завершаем алгоритм. Получаем для примера код Прюфера (1,2,1,3,3,5).
Постройте код Прюфера для дерева:

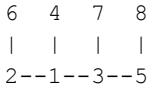


Ответ запишите в таком же виде, как в примере (то есть последовательность чисел записывается в скобках, числа разделяются запятой, пробелов в ответе быть не должно).

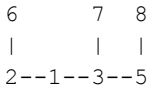
Задача 3: Код Прюфера - 2

Кодирование Прюфера переводит помеченные деревья порядка n в последовательность чисел от 1 до n по алгоритму: Пока количество вершин больше двух:
1) Выбирается лист v с минимальным номером.

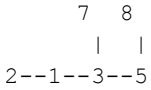
2) В код Прюфера добавляется номер вершины, смежной с v.
3) Вершина v и инцидентное ей ребро удаляются из дерева.
Полученная последовательность называется кодом Прюфер для заданного дерева.
Рассмотрим построение кода Прюфера на примере. Возьмем дерево:



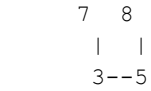
Наименьший лист в дереве 4, номер смежной вершины 1, поэтому в код Прюфера добавляем (1).



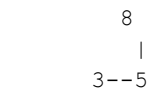
На следующем шаге Наименьший лист в дереве 6, номер смежной вершины 2, поэтому в код Прюфера добавляем 2 и получаем (1,2).



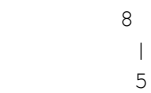
(1,2,1)



(1,2,1,3)



(1,2,1,3,3)



(1,2,1,3,3,5)
(Осталось 2 вершины, завершаем алгоритм. Получаем для примера код Прюфера (1,2,1,3,3,5).
Напишите программу, которая для заданного кода Прюфера из n-2 чисел от 1 до n восстанавливает дерево
На ввод подается N-2 целых чисел в диапазоне от 1 до N.
Выведите все ребра дерева как пары вершин, которые являются концами ребра. Вершины в ребре необходимо упорядочить по возрастанию, список ребер отсортируйте.

Примеры

Входные данные

1 2 1 3 3 5

Выходные данные

1 2
1 3
1 4
2 6
3 5
3 7
5 8

Задача 4: Последовательность - 2

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = (a_{n-1} + a_{n-2}) \bmod K$,
$a_1 = A$,
$a_2 = B$,
Где mod - операция взятия остатка деления.
Если выписать всю последовательность, то в какой-то момент числа в ней начнут повторяться. Ваша задача для заданных A, B, K ($0 < A, B, K < 1000$) определить длину цикла в последовательности.
Длиной цикла c называется такое минимальное число $p > 0$, что существует некоторый начальный номер элемента последовательности n0, что для любого $n \geq n0$ выполняется $a_n = a_{n+p}$.
На стандартный поток вводятся три числа A, B и K. Ответ выводится на стандартный поток вывода. Числа на стандартном потоке ввода разделяются произвольным количеством пробельных символов (переводов строк, пробелов и табуляций).

Примеры

Входные данные

1 1 10

Выходные данные

60

Задача 5: Сжатие текста

Важный раздел информатики - это алгоритмы сжатия данных. В этой задаче мы будем рассматривать только сжатие английских текстов без потерь. Задача заключается в том, что входной текст преобразовываются в закодированные бинарные данные таким образом, чтобы в размер сжатых данных был минимальным. Разработайте алгоритм и напишите программу, которая

реализовывает наиболее эффективное по вашему мнению алгоритм сжатия и распаковки *естественных* текстов на английском языке.
На вход программе подается либо текст для сжатия, либо поток данных для распаковки, а именно: на стандартном потоке ввода (т. е., например, с клавиатуры) программе сначала задается число 0, если программа должна сжать данные, либо число 1, если программа должна распаковать данные.
В случае сжатия далее на вход подается текст состоящий из символов с кодами ascii до 127.
В случае сжатия данных программа должна вывести на стандартный поток вывода (экран) закодированные данные как строка из '0' и '1' и завершить работу
В случае декодирования данных программа должна вывести на стандартный поток вывода исходный текст.
Программа в режиме распаковки данных должна распаковывать данные, сжатой той же самой вашей программой в режиме закодирования данных. Никаких других данных для распаковки подаваться не будет.
Ваша задача - придумать и реализовать такой алгоритм сжвтия и распаковки, который имел бы наилучшую эффективность, то есть размер сжатого текста.
Тестирование будет производиться на наборе тестовом и контрольных наборах. Баллы за каждый текст будут выставляться в зависимости от степени сжатия
Пример входных данных для кодирования

```
0
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
```

Возможный пример результата работы:

```
01100101 01001110 01110000 01111010 01111010 01000011 01110011 01110101
01010100 01111001 00110001 01010011 01001011 01001101 01101100 01011000
01001011 01001101 01101100 01001001 01010110 01010001 01101010 01001110
01001011 01100011 01101110 01001101 01010100 01010011 01111000 01001010
01010110 01010001 01100111 01110011 01010100 01010011 00110000 01110101
01111001 01100011 01111010 01010000 01010101 00111000 01101000 01010000
01010101 00101111 01000100 01001010 01010100 01000101 01110110 01010110
01100111 01010101 01101010 01101110 01011010 01011010 01100001 01101100
01000110 01101000 01010101 01000100 01100101 01011001 01101100 00110101
01001011 01010001 01110001 01110101 01010001 01001000 01011010 01101100
01010011 01010101 01011010 01101101 01011000 01101010 01101111 01110110
01101100 00101011 01001111 01101111 01001101 01100001 01010000 01000111
01000100 01001010 01010001 01111000 01000001 01001100 01010000 01101111
00101011 01111001 01000001 00111101
```

Пример входных данных для декодирования:

1
01100101010011100111000001111010011110100100001101110011011101010101010000111
10010011000101010011010010110100110101101100010110000100101101001101011011 00
01001001010101100101000101101010010011100100101101100011011011100100110101 01
01000101001101111000010010100101011001010001011001110111001101010100010100 11
00110000011101010111100101100011011110100101000001010101001110000110100001 01
00000101010100101111010001000100101001010100010001010111011001010110011001 11
01010101011010100110111001011010010110100110000101101100010001100110100001 01
01010100010001100101010110010110110000110101010010110101000101110001011101 01
01010001010010000101101001101100010100110101010101011010011011010101100001 10
10100110111101110110011011000010101101001111011011110100110101100001010100 00
01000111010001000100101001010001011110000100000101001100010100000110111100 10
10110111100101000001001111 01

Результат работы:

Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything
Answer to the Ultimate Question of Life, the Universe, and Everything

Задача 6: Художник-3

Для создания новой картины мальчик Казимир взял клечатый лист бумаги размера 6*N. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы она соприкасалась ровно в одной точке ровно с одной из уже покрашенных. Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется оценить число вариантов картин, которые могли получиться.
На стандартном потоке вводится число N < 101
Выведите одно число - количество различных картин, которые могли получиться по модулю 1000000007

Примеры

Входные данные

2

Выходные данные

2

Задача 7: Befunge

Befunge — стековый эзотерический язык программирования. Считается двумерным, так как программа на Befunge записывается в таблицу со сшитыми краями (тор), по которой в различных направлениях перемещается интерпретатор, исполняя команды, расположенные в её ячейках. Размер таблицы ограничен 25 строками и 80 столбцами.
Ниже перечислены команды языка Befunge. Каждая команда кодируется одним ASCII-символом. Команды, берущие параметры из стека, удаляют их со стека.

перемещение (9):
> Двигаться вправо
< Двигаться влево
^ Двигаться вверх
v Двигаться вниз
_ Двигаться вправо, если на вершине стека 0, иначе — влево.
| Двигаться вниз, если на вершине стека 0, иначе — вверх.
? Двигаться в случайном направлении
Пропустить следующую ячейку ("трамплин")
@ Конец программы
манипулирование со стеком (3):
: Поместить в стек копию вершины
\ Обменять местами вершину и подвершину
$ Удалить вершину
модификация кода программы (2):
p "PUT": со стека извлекаются координаты ячейки и ASCII-код символа, который помещается по этим координатам
g "GET": со стека извлекаются координаты ячейки; ASCII-код символа по этим координатам помещается в стек
константы (2):
0-9 Поместить число в стек
" Начало/конец символьного режима, в котором ASCII-коды всех текущих символов программы помещаются в стек
стековые арифметические операции (5):
+ Сложение вершины и подвершины
- Вычитание вершины и подвершины
* Умножение вершины и подвершины
/ Целочисленное деление
% Остаток от деления
стековые логические операции (2):
! Отрицание: нуль на вершине заменяется на 1, ненулевое значение — на 0
` Сравнение "больше, чем": если подвершина больше вершины, в стек помещается 1, иначе 0 (forth:>)
ввод-вывод (4):
& Запросить у пользователя число и поместить его в стек
~ Запросить у пользователя символ и поместить в стек его ASCII-код
. Распечатать вершину стека как целое число
, Распечатать символ, соответствующий ASCII-коду на вершине стека

Вам требуется написать интерпретатор этого языка прогрммирования.
В первой строке задается строка с входными данными программы. В последующих строках программа на языке Befunge.
Выведите результат работы программы. Считайте, что программа не зацикливается.

Примеры

Входные данные

```
1
62*1+v>01p001>+v>\:02p\:02gv
     0       ^             <
     .          :p
     "          .1
        v 0," "<0
     "  >1g12-+:|
     ,           @
     >^
```

Выходные данные

0 1 1 2 3 5 8 13 21 34 55 89 144 233

Задача 8: Звездчатый многоугольник

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

Требуется написать программу, которая по заданному набору вершин строит звездчатый многоугольник состоящий из этих вершин

На стандартном потоке ввода задано число вершин множества ($2 < N < 1000000$). Затем идут N пар целых чисел x_i, y_i - координаты точек. Все координаты не превышают по модулю 10000.
На стандартный поток вывода выведите вершины многоугольника в порядке обхода против часовой стрелки.

Примеры

Входные данные

```
6
0 0
1 1
1 2
2 0
2 1
0 2
```

Выходные данные

0 0
2 0
2 1
1 1
1 2
0 2

Олимпиада "Ломоносов" по информатике. 2016-17.
5-9 класс. Отборочный тур.

Задача 1.1: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 1$,
$a_2 = 1$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 114 по 187 символ (символы нумеруются с единицы).

Задача 1.2: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 0$,
$a_2 = 1$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 124 по 178 символ (символы нумеруются с единицы).

Задача 1.3: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,
$a_1 = 1$,
$a_2 = 0$,
$a_3 = 1$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 102 по 165 символ (символы нумеруются с единицы).

Задача 1.4: Последовательность

Последовательность целых чисел $\{a_n\}$ задается следующим рекуррентным соотношением:
$a_n = 2 * a_{n-3} - a_{n-2} + a_{n-1}$,

$a_1 = 1$,
$a_2 = 1$,
$a_3 = 0$.
Все числа последовательности выписали в строку. Отправьте на проверку число с 110 по 196 символ (символы нумеруются с единицы).

Задача 2: Запись числа

На вход программе подаётся положительное целое число N (N < 1000001) и последовательность из N символов. Требуется определить, можно ли из введённой последовательности выбрать символы (в любом порядке) так, чтобы получилась запись шестнадцатеричного числа по правилам языка Си. В языке Си запись шестнадцатеричного числа начинается с нуля (0), за ним следует строчная буква икс (x), за которой следуют одна или более чем одна шестнадцатеричная цифра (любая из: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Составлять запись числа можно только из тех “буквенных” цифр, которые подходят по регистру, то есть, являются заглавными. Любая цифра в записи числа может быть использована не большее количество раз, чем количество её вхождений во введенную последовательность. Если можно составить запись хотя бы для одного числа, то нужно вывести запись для наибольшего из возможных чисел. Если запись составить нельзя, программа выводит No

Примеры

Входные данные

17
I-AM-A-PROGRAMMER

Выходные данные

No

Входные данные

17
1xAMxAxPR0GRaMMER

Выходные данные

0xEAA1

Задача 3.1: Художник

Для создания новой картины мальчик Казимир взял клечатый лист бумаги размера 3*6. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

01
11

10

11

11
10

11
01

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.2: Художник

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 4*5. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

01
11

10
11

11
10

11
01

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.3: Художник

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 6*3. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

01
11

```
10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 3.4: Художник

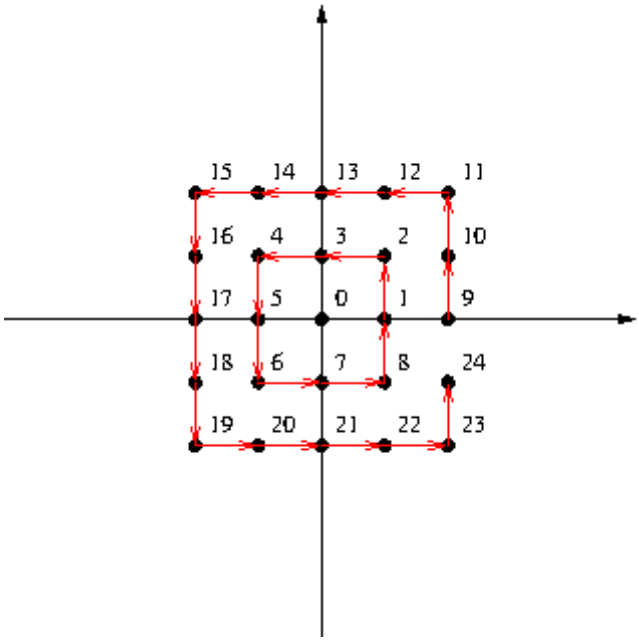
Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 5*4. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется получить все варианты картин, которые могли получиться.
Например на поле 2x2 могло получиться 4 картины

```
01
11

10
11

11
10

11
01
```

Ответ отправьте в формате как в примере. Картины разделяются пустой строкой. Покрашенная клетка выводится символом 1, пустая 0.

Задача 4: Прямые и точки

Рассмотрим координатную плоскость. Занумеруем точки с целочисленными координатами следующим образом. Точка (0,0) получает номер 0. Точка (1,0) получает номер 1, (1,1) получает номер 2, (0,1) получает номер 3 и так далее против часовой стрелки.



Напишите программу, определяющую принадлежность точек прямой. Сначала на стандартном потоке ввода задаются два номера точек, определяющие прямую на плоскости. Точки не совпадают. Затем задается последовательность номеров точек, принадлежность которых заданной прямой требуется проверить. Последовательность заканчивается числом -1.
Для каждой точки в последовательности на стандартный поток вывода напечатайте 0, если эта точка не находится на заданной прямой, и 1, если точка находится на прямой.
Все номера точек - 32-битные положительные целые числа.
Число проверяемых точек во входном потоке не превышает 500000.

Примеры

Входные данные
16 2 12 3 6 -1

Выходные данные
0 1 0

Задача 5: Сжатие изображения

Важный раздел информатики - это алгоритмы сжатия изображений. В этой задаче мы будем рассматривать только сжатие черно-белых картинок без потерь. Задача помехоустойчивого кодирования заключается в том, что входные бинарные данные преобразовываются в закодированные бинарные данные таким образом, чтобы в размер сжатых данных был минимальным. Разработайте алгоритм и напишите программу, которая реализовывает наиболее эффективное по вашему мнению алгоритм сжатия и распаковки черно-белых изображений.

На вход программе подается либо изображение для сжатия, либо поток данных для распаковки, а именно: на стандартном потоке ввода (т. е., например, с клавиатуры) программе сначала задается число 0, если программа должна сжать данные, либо число 1, если программа должна распаковать данные.
В случае кодирования далее на вход подается картинка 51 строк по 100 символов. Цифра ‘0’ соответствует белому цвету, цифра ‘1’ соответствует черному. Все прочие символы (пробелы, переводы строк) должны игнорироваться.
В случае сжатия данных программа должна вывести на стандартный поток вывода (экран) закодированные данные как строка из '0' и '1' и завершить работу
В случае декодирования данных программа должна вывести на стандартный поток вывода (экран) изображение из 51 строк по 100 символов '0' или '1'.
Программа в режиме распаковки данных должна распаковывать данные, сжатой той же самой вашей программой в режиме закодирования данных. Никаких других данных для распаковки подаваться не будет.
Ваша задача - придумать и реализовать такой алгоритм сжвтия и распаковки, который имел бы наилучшую эффективность, то есть размер сжатой картинки. Тестирование будет производиться на наборе тестовом и контрольных наборах. Баллы за каждое изображение будут выставляться в зависимости от степени сжатия
Пример входных данных для кодирования (в условии размер картинки для простоты 3x3):

```
0
010
000
000
```

Возможный пример результата работы:

```
0001
```

Пример входных данных для декодирования:

```
1
0001
```

Результат работы:

```
010
000
000
```

Задача 6: Художник-2

Для создания новой картины мальчик Казимир взял клечаты лист бумаги размера 4*N. Затем он выбрал произвольную клетку и покрасил её в черный цвет. Каждую следующую клетку он красил так, чтобы у нее была ровно одна

покрашенная соседняя клетка (по сторонам). Так продолжалось до тех пор, пока были клетки котые можно покрасить по этому правилу.
Вы решили повторить работу Казимира и вам требуется оценить число вариантов картин, которые могли получиться.
На стандартном потоке вводится число N < 10001
Выведите одно число - количество различных картин, которые могли получиться по модулю 1000000007

Примеры

Входные данные
2

Выходные данные
10

Задача 7: Анти-Бармаглот

Компания *Barmaley's Computing* учла пожелания пользователей и выпустила вторую версию компьютера - *Barmaglot-2*. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер остался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека могут содержать либо число, либо латинскую букву или пробел, либо специальное значение [BARMALEY], которое используется для обозначения логической лжи, при том что любое другое обозначает истину.

Программа для Barmaglot'a 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 латинские буквы означают "занести данную букву в регистр".
- Команды 0, 1, ..., 9 означают "занести в регистр соответствующее число".
- Команда @ означает занести в регистр пробел.

- Команды +, -, *, /, % означают соответствующие арифметические действия, операции целочисленные. % - операция взятия остатка
- Команды <, >, = означают сравнение двух чисел или двух символов, & и | означают логическое "и" и логическое "или";

все эти команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.

- Команда # умножает содержимое регистра в десять раз,
- Команда _ делит содержимое регистра в десять раз.
- Команда ! работает как логическое отрицание содержимого регистра: если там значение [BARMALEY], то заносится значение 1, если любое другое - заносится значение [BARMALEY].
- Команда . выдаёт текущее значение регистра на печать
- команда ? вводит очередное число(целиком, а не по разрядам), а если на вводе кончилась (оборвалась) лента, заносит в регистра значение [BARMALEY].
- Команда] заносит значение из регистра в стек;
- Команда [извлекает значение с вершины стека и заносит его в регистра (если извлекать нечего, в регистр заносится [BARMALEY]);
- Команда ~ меняет местами значения в регистра и на вершине стека.
- Команда } заносит значение из регистра в очередь,
- Команда { извлекает из очереди самое старое значение и помещает в аккумулятор (или помещает туда [BARMALEY], если очередь пуста).
- Команды (и) предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Команда (, если в регистре [BARMALEY], идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в регистре было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда), наоборот, если в регистре [BARMALEY], не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной круглой 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.

Пример программы, которая печатает традиционную строчку “HELLO WORLD”:
H.E.L.L.O.@.W.O.R.L.D."

Вам требуется написать программу которая эмулирует программу на этом языке программирования.

В первой строке вводится программа для Barmaglot. Во второй строке исходное содержимое ленты.

Требуется вывести результат работы программы или "[ERROR]" если выполнение завершилось с ошибкой или программа синтаксические неверная.

Пример работы эмулятора можно найти по ссылке
http://ejudge.cs.msu.ru/barmaglot2/

Примеры

Входные данные
]?(}}?)}!{{(]]{(~]{>([}}!])[(~}}!)[]]{()}[.@.{{)"
2 10 3 9

Выходные данные
2 3 9 10

Входные данные
H.H.
text

Выходные данные
[ERROR]

Задача 8: Звездчатый многоугольник

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

Требуется написать программу, которая по заданному звездчатому многоугольнику найти его ядро. Можно заметить что ядро звездчатого многоугольника тоже многоугольник

На стандартном потоке ввода задано число вершин звездного многоугольника ($2 < N < 20$). Затем идут N пар целых чисел x_i, y_i - координаты вершин многоугольника в порядке обхода против часовой стрелки. Все координаты не превышают по модулю 10000.
На стандартный поток вывода выведите число вершин ядра исходного многоугольника, затем вершины ядра против часовой стрелки в том же формате, что и во вводе. Среди всех возможных ответом выведите тот, в котором наименьшее число вершин.

Примеры

Входные данные

```
6
0 0
2 0
2 1
1 1
1 2
0 2
```

Выходные данные

```
4
0 0
1 0
1 1
0 1
```

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

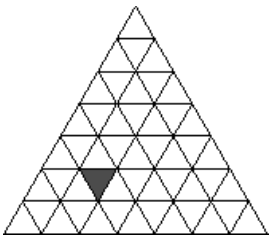
Замечание

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%

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

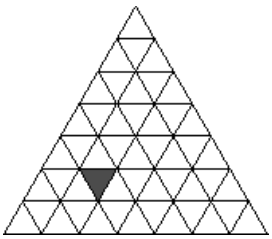
Замечание

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

Ответы и критерии проверки

Тесты, проверяющие программы и ответы находятся в архиве тестирующей системы:
https://ejudge.cs.msu.ru/lom17.tgz
Протоколы проверки и технические баллы доступны по логину и паролю участника
https://ejudge.cs.msu.ru/cgi-bin/new-client?contest_id=73
https://ejudge.cs.msu.ru/cgi-bin/new-client?contest_id=74

A-1) Результирующий файл в архиве A-1-1/tests/001.ans и A-1-2/tests/001.ans
A-2) Результирующий файл ~1ГБ, проверяемая часть файла в архиве (A-2-1 и A-2-2)
A-3) Программа в архиве A-3/sol.cpp

B) Тесты и проверяющая программа в архиве - B-2/tests. В задаче требовалось раскрасить поле в шахматном порядке (угловые клетки белые) и для черных начальных клеток выигрышный ход в соседнюю. Для белых - вывести 0

C) Можно было решить методом динамического программирования по профилю. Тесты и решения в архиве.

D) Тесты и проверяющая программа в архиве. Баллы выставлялись в зависимости от полноты расшифровки.

E) Ответы в архиве в соответствующих подзадаче директориях

F) Ответы и тесты в архиве.

Каждая задача оценивается в 100 технических баллов. Разделение по подзадачам:

A	20/30/50
B	30/70
C	30/70
D	20/30/50
E	10/20/20/50
F	30/70

5. Сведения об опубликованных сборниках олимпиадных заданий и методических пособиях

нет публикаций.