

Задача А. Линейный конгруэнтный генератор

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Как известно, линейный конгруэнтный генератор псевдослучайных чисел устроен следующим образом. Зафиксированы числа a , c и m . Кроме того, у генератора есть состояние s . Когда нужно сгенерировать очередное псевдослучайное число, происходит присваивание $s_{\text{new}} \leftarrow (s_{\text{old}} \cdot a + c) \bmod m$, после чего новое значение s объявляется ответом.

В этой задаче $m = 2^{32}$, а числа a , c и начальное состояние s заданы во входных данных. Выведите следующие 10 чисел, которые сгенерирует этот генератор.

Формат входных данных

В первой строке записаны через пробел три целых числа: a , c и s . Гарантируется, что эти числа помещаются в 32-битный беззнаковый тип данных.

Формат выходных данных

Выведите 10 следующих чисел, которые сгенерирует получившийся генератор.

Пример

<i>стандартный ввод</i>
69069 1 12345
<i>стандартный вывод</i>
852656806 3856338159 1023442532 1580485141 1639408594 4089354539 1989334640 1055483825 2732393918 2852536103

Задача В. Разворот линейного конгруэнтного генератора

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Рассмотрим следующий линейный конгруэнтный генератор псевдослучайных чисел. У генератора есть состояние — целое число s ($0 \leq s < 2^{32}$). Каждый раз, когда требуется очередное псевдослучайное число, генератор выполняет преобразование $s_{\text{new}} = (a \cdot s_{\text{old}} + c) \bmod m$. После этого s_{new} становится новым состоянием генератора, а также возвращается в качестве псевдослучайного числа.

В этой задаче $a = 69\,069$, $c = 1$ и $m = 2^{32}$ (такие значения использовались в старых версиях библиотеки glibc).

Генератор был приведён в некоторое начальное состояние, после чего сгенерировал одно псевдослучайное число. Это число известно. Восстановите начальное состояние генератора.

Формат входных данных

В единственной строке задано целое число v — сгенерированное число ($0 \leq v < 2^{32}$).

Формат выходных данных

Выведите одно число: начальное состояние генератора.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1	0
2	2783094533

Пояснения к примерам

В первом примере $(0 \cdot 69\,069 + 1) \bmod 2^{32} = 1$.

Во втором примере $(2\,783\,094\,533 \cdot 69\,069 + 1) \bmod 2^{32} = 2$.

Задача С. Известная задача

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Это интерактивная задача.

Маленькая Аня занимается спортивным программированием. Она тоже хочет готовить задачи для соревнований!

Аня пришла к координатору олимпиады и сказала, что хочет сделать простую задачу: посчитать сумму двух чисел. Конечно, координатор ответил, что такая задача уже есть.

Аня подумала и предложила просуммировать не два числа, а 10 000 чисел. К сожалению, и на это координатор ответил, что подобная задача уже используется на пробном туре.

Аня ещё подумала и предложила просуммировать 100 000 000 чисел. Координатор задумался, но потом заметил, что тесты получатся слишком большие, и олимпиадный сервер не справится с нагрузкой.

Тогда Аня предложила сделать задачу интерактивной, а тесты не хранить, а генерировать на ходу. Координатор заинтересовался и спросил, как именно будут генерироваться тесты.

Тут Аня вспомнила про RANDU — один из первых генераторов псевдослучайных чисел, получивших широкое распространение, — и предложила такой способ. Пусть жюри загадает целое число s от 0 до $2^{31} - 1$, которое держится в секрете, но зафиксировано заранее для каждого теста. Чтобы получить x , очередное число последовательности, заменим s на $(s \cdot 65\,539) \bmod 2^{31}$, после чего дадим участнику число $x = s \bmod 1\,000\,000$. Задача участника — найти сумму всех значений x , которые ему будут даны.

Координатор подумал и согласился, только попросил добавить в протокол возможность вывести ответ раньше, чем прочитаны все числа. Аня задумалась: зачем может понадобиться такая возможность? Но тоже согласилась.

Решите задачу Ани в общем виде. Получая последовательность из n псевдослучайных чисел, сгенерированных, как описано выше, найдите сумму всех этих чисел.

Протокол взаимодействия

Сначала на вход подаётся целое число n в отдельной строке ($1 \leq n \leq 100\,000\,000$). Далее участник задаёт от 0 до n вопросов, после чего выводит ответ.

Чтобы задать очередной вопрос, в отдельной строке выведите «? t », где t — сумма чисел последовательности, полученных на данный момент.

Чтобы вывести ответ, в отдельной строке выведите «! a », где a — сумма всех n чисел последовательности. После этого корректно завершите работу программы.

Чтобы предотвратить буферизацию вывода, после вывода каждой строки следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>	<i>секретное число</i>
4	? 0	($s = 238\,572\,887$)
5	? 5	($s = 5$)
327695	? 327700	($s = 327\,695$)
966125	? 1293825	($s = 1\,966\,125$)
847495	! 2141320	($s = 8\,847\,495$)

Задача D. Камень, ножницы, бумага на плоскости

Имя входного файла: *стандартный ввод*
 Имя выходного файла: *стандартный вывод*
 Ограничение по времени: 1 секунда
 Ограничение по памяти: 512 мегабайт

В этой задаче нужно пройти из одного угла клетчатого поля в другой, следуя определённым правилам и сделав не слишком много ходов.

На плоскости нарисовано клетчатое поле, состоящее из m рядов и n столбцов. В каждой клетке этого поля лежит один из трёх предметов: камень, ножницы или бумага. Мы начинаем движение в клетке $(1, 1)$ и хотим добраться до клетки (m, n) , последовательно делая ходы. За один ход можно переместиться в любую клетку на расстоянии не больше d от текущей. Расстояние измеряется между центрами клеток: клетки (x_1, y_1) и (x_2, y_2) находятся друг от друга на расстоянии $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. При этом каждый ход должен вести в клетку с предметом, побеждающим предмет на предыдущей посещённой клетке. Как и в игре «Камень, ножницы, бумага», камень побеждает ножницы, ножницы побеждают бумагу, а бумага побеждает камень. Когда мы оказываемся в какой-то клетке, мы забираем из неё предмет, поэтому перемещаться дважды в одну и ту же клетку нельзя.

Поле довольно большое, поэтому типы предметов в клетках задаются с использованием генератора случайных чисел, а точнее, линейного конгруэнтного генератора. У этого генератора есть состояние — целое число s от 0 до $2^{32} - 1$. Чтобы получить очередное случайное число от 0 до $r - 1$, производится присваивание $s := (s \cdot 1\,664\,525 + 1\,013\,904\,223) \bmod 2^{32}$ и возвращается остаток от деления s на r . Начальное значение s задано во вводе.

Предметы в клетках определяются следующим образом. Для каждой клетки получается очередное случайное число из множества $\{0, 1, 2\}$ (то есть $r = 3$). Если это число равно нулю, предмет оказывается камнем, если единице — бумагой, а если двойке — ножницами. Клетки рассматриваются по рядам сверху вниз, а клетки в каждом ряду — слева направо. Таким образом, предметы в клетках определяются в следующем порядке: $(1, 1)$, $(1, 2)$, \dots , $(1, n)$, $(2, 1)$, $(2, 2)$, \dots , $(2, n)$, \dots , $(m, 1)$, $(m, 2)$, \dots , (m, n) .

Важное дополнительное условие состоит в том, что нужно проделать весь путь за время, не превосходящее время, за которое можно было бы прийти из клетки $(1, 1)$ в клетку (m, n) по прямой с половиной максимальной скорости, плюс ещё три дополнительных хода. Формально количество ходов не должно превосходить вещественного числа $2 \cdot \frac{\sqrt{(m-1)^2 + (n-1)^2}}{d} + 3$.

По заданным размерам поля и максимальному расстоянию для каждого хода, а также начальному состоянию генератора случайных чисел, найдите путь, удовлетворяющий всем ограничениям.

Формат входных данных

В первой строке ввода заданы через пробел целые числа m , n , d и s — высота и ширина поля, максимальное расстояние для каждого хода, а также начальное состояние генератора случайных чисел ($100 \leq m, n \leq 2^{16}$, $10 \leq d \leq 100$, $0 \leq s \leq 2^{32} - 1$). Обратите внимание на то, что числа m , n и d ограничены снизу достаточно большими значениями.

Формат выходных данных

В первой строке выведите количество ходов k в найденном пути. Далее в $(k+1)$ строке выведите координаты посещённых клеток — сначала ряд, затем столбец — в порядке их посещения.

Первой должна быть посещена клетка $(1, 1)$, последней — клетка (m, n) . Никакая клетка не должна быть посещена больше одного раза. Предмет в каждой следующей клетке должен побеждать предмет в предыдущей клетке. Расстояние между любыми двумя соседними клетками пути не должно превосходить d , а общее количество ходов k не должно превосходить вещественного числа $2 \cdot \frac{\sqrt{(m-1)^2 + (n-1)^2}}{d} + 3$.

Если возможных решений несколько, выведите любое из них. Гарантируется, что во всех тестах жюри существует путь, удовлетворяющий всем ограничениям, длина которого не превосходит вещественного числа $1.5 \cdot \frac{\sqrt{(m-1)^2 + (n-1)^2}}{d} + 3$.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
100 120 50 5	5 1 1 31 41 59 70 92 107 98 120 100 120

Пояснение к примеру

В нижеследующей таблице приведены значения s и предметы для клеток, посещённых в примере.

клетка	значение s	$s \bmod 3$	предмет
(1, 1)	1022226848	2	ножницы
(31, 41)	4062427704	0	камень
(59, 70)	11774611	1	бумага
(92, 107)	860629922	2	ножницы
(98, 120)	383195253	0	камень
(100, 120)	2533494757	1	бумага

Значение дроби $\frac{\sqrt{(m-1)^2+(n-1)^2}}{d}$ равно $3.0959\dots$, поэтому в примере нужно сделать не более девяти ходов, а жюри формально гарантирует, что существует путь длиной не более семи ходов.

Задача Е. Создай лучшего питомца

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Петя играет в многопользовательскую компьютерную игру. Герой, за которого играет Петя, может иметь до трёх питомцев.

У каждого питомца есть восемь параметров, отвечающих за силу его атаки и защиты. Каждый параметр — это целое число от 1 до 999. Числа больше 500 влияют в первую очередь на силу атаки, а меньшие — на силу защиты. В целом, чем больше каждый параметр отличается от 500 в любую сторону, тем лучше. Кроме того, у каждого питомца есть восемь генетических параметров, отвечающих за его внешний вид. Наконец, каждый питомец имеет имя.

При покупке питомца в игровом магазине можно выбрать для него только имя. Дальше запускается алгоритм, который выбирает силовые и генетические параметры случайным образом. Петя уже приобрёл одного питомца, но остался недоволен получившимся: силовые параметры оказались не очень хорошими. Он твёрдо решил, что следующий питомец должен быть сильным, и не остановится ни перед чем для достижения этой цели. Но как этого добиться?

После некоторых исследований Пете удалось раздобыть кусок кода на скриптовом языке XuzzyLang, отвечающий за генерацию параметров. Вот этот кусок:

```
CreateRandomPet (Name):  
    Seed := Now xor Hash (Name) xor Salt  
    return Pet:  
        Pet.Name := Name  
        for I := 1, 2, ..., 8:  
            Pet.Power[I] := GenerateRandomPower  
        for J := 1, 2, ..., 8:  
            Pet.Gene[J] := Random mod 5  
  
Hash (String):  
    Result := 0  
    for I := 1, 2, ..., String.Length:  
        Result := Result * 31 + String[I].AsciiCode  
    return Result  
  
Random:  
    Seed := Seed * 1234567893 + 151515151  
    return Seed  
  
GenerateRandomPower:  
    Result := 500  
    for I := 1, 2, ..., 10000:  
        Result := Result + Random mod 3 - 1  
    return Result
```

Петя раньше не видел программ на этом языке, но успел выяснить, что все числа в программе целые, а все вычисления производятся по модулю 2^{31} . Как видно из кода, параметры питомца полностью определяются тремя числами: Now, Hash(Name) и Salt. Петя знает, что функция Now возвращает текущий момент в формате unix timestamp (количество секунд, прошедших с 1 января 1970 года). Функция Hash использует ASCII-коды символов (65–90 для A–Z и 97–122 для a–z). А вот число Salt Пете узнать не удалось.

Петя считает, что качество питомца — это среднее отклонение силовых параметров от числа 500, генетические параметры ему не важны.

Петя уже придумал имя для своего питомца, и хочет создать его в момент Start. Однако, чтобы питомец получился сильным, он готов подождать от 0

до Wait секунд включительно. Выясните при этих условиях, в какой именно момент следует совершить покупку, чтобы получить питомца с максимальным значением качества.

Во всех тестах в этой задаче число Salt равно одному и тому же секретному значению от 0 до $2^{31} - 1$. В примере приведён первый питомец, приобретённый Петей.

Формат входных данных

В первой строке записаны два целых числа Start и Wait, за которыми следует слово Name. ($1\,535\,814\,000 \leq \text{Start} \leq 2\,000\,000\,000$, $0 \leq \text{Wait} \leq 120$). Слово Name имеет длину от 1 до 20 символов и может содержать только английские буквы в любом регистре.

Формат выходных данных

В первой строке выведите сначала целое число — момент генерации наилучшего питомца, а затем вещественное число — качество этого питомца. Выведите это число как можно более точно. Во второй строке выведите восемь чисел — значения силы сгенерированного питомца в порядке их следования. В третьей строке выведите восемь чисел — генетические параметры питомца в порядке их следования.

Если возможных питомцев с максимальным значением качества несколько, выведите того из них, которого можно сгенерировать раньше других.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
1535814000 0 Murik	1535814000 69.250 484 467 469 363 621 504 291 503 1 3 2 3 0 0 4 2

Задача F. Повторное перемешивание

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Рассмотрим следующий линейный конгруэнтный генератор случайных чисел. У генератора есть состояние `state`, которое может быть любым целым числом от 0 до $2^{32} - 1$ включительно. Чтобы сгенерировать следующее случайное целое число от 0 до `range - 1` включительно, используется функция `random`:

```
Function random (range):  
    state := (state * 1664525 + 1013904223) mod 232;  
    return (state * range) div 232;
```

Здесь `mod` и `div` — операции взятия остатка по модулю и целочисленного деления. Выбранные константы рекомендованы в книге *Numerical Recipes*. К примеру, если в какой-то момент `state` равно 12345, то при вызове `random (100)` сначала `state` становится равным 87628868, а затем функция возвращает значение 2.

Чтобы случайным образом перемешать массив, используется функция `shuffle`:

```
Function shuffle (array):  
    n := length of array;  
    for i := 0, 1, 2, ..., n - 1:  
        j := random (i + 1);  
        array[i] <-> array[j];  
    return array;
```

Здесь операция «`x <-> y`» меняет местами элементы `x` и `y`; если это один и тот же элемент массива, ничего не происходит. Массив индексируется целыми числами от 0 до $n - 1$ включительно. К примеру, если в какой-то момент `state` равно 12345, то при вызове `shuffle` от массива `[1, 2, 3, 4, 5]` итоговое значение `state` будет равно 3908547000, а функция вернёт массив `[2, 3, 4, 1, 5]`.

Фёдор выбрал целое число `seed` от 0 до $2^{32} - 1$ включительно. После этого он выполнил следующую программу:

```
state := seed;  
n := 10000;  
array := [1, 2, ..., n];  
array := shuffle (array);
```

Фёдор держит в секрете значение `seed`, однако раскрыл содержимое массива `array` после выполнения этой программы. Выясните, что получится у Фёдора в массиве `array`, если он добавит в конец своей программы ещё одну строчку:

```
array := shuffle (array);
```

Формат входных данных

В первой строке записано целое число n — количество элементов в массиве (в этой задаче это число всегда равно 10000). Во второй строке задана перестановка из n чисел от 1 до n через пробел — содержимое массива после выполнения программы Фёдора.

Формат выходных данных

В первой строке выведите перестановку из n чисел от 1 до n через пробел — содержимое массива после выполнения программы с добавленной строчкой.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
10000 7236 5734 5374 ... 796 5348	8150 6681 695 ... 5947 4461

Пояснение к примеру

В примере значение `seed` равно 12345. Полностью пример можно скачать по адресу:

<https://tswb.ru/trains/shuffle-again>.

Задача G. Двойное перемешивание

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 2 секунды
Ограничение по памяти: 512 мегабайт

Рассмотрим следующий линейный конгруэнтный генератор случайных чисел. У генератора есть состояние `state`, которое может быть любым целым числом от 0 до $2^{32} - 1$ включительно. Чтобы сгенерировать следующее случайное целое число от 0 до `range - 1` включительно, используется функция `random`:

```
Function random (range):  
    state := (state * 1664525 + 1013904223) mod 232;  
    return (state * range) div 232;
```

Здесь `mod` и `div` — операции взятия остатка по модулю и целочисленного деления. Выбранные константы рекомендованы в книге *Numerical Recipes*. К примеру, если в какой-то момент `state` равно 12345, то при вызове `random (100)` сначала `state` становится равным 87628868, а затем функция возвращает значение 2.

Чтобы случайным образом перемешать массив, используется функция `shuffle`:

```
Function shuffle (array):  
    n := length of array;  
    for i := 0, 1, 2, ..., n - 1:  
        j := random (i + 1);  
        array[i] <-> array[j];  
    return array;
```

Здесь операция «`x <-> y`» меняет местами элементы `x` и `y`; если это один и тот же элемент массива, ничего не происходит. Массив индексируется целыми числами от 0 до `n - 1` включительно. К примеру, если в какой-то момент `state` равно 12345, то при вызове `shuffle` от массива [1, 2, 3, 4, 5] итоговое значение `state` будет равно 3908547000, а функция вернёт массив [2, 3, 4, 1, 5].

Фёдор выбрал целое число `seed` от 0 до $2^{32} - 1$ включительно. После этого он выполнил следующую программу:

```
state := seed;  
n := 10000;  
array := [1, 2, ..., n];  
array := shuffle (array);  
array := shuffle (array);
```

Фёдор держит в секрете значение `seed`, однако раскрыл содержимое массива `array` после выполнения этой программы. Выясните, что получится у Фёдора в массиве `array`, если он добавит в конец своей программы ещё одну строчку:

```
array := shuffle (array);
```

Формат входных данных

В первой строке записано целое число `n` — количество элементов в массиве (в этой задаче это число всегда равно 10000). Во второй строке задана перестановка из `n` чисел от 1 до `n` через пробел — содержимое массива после выполнения программы Фёдора.

Формат выходных данных

В первой строке выведите перестановку из `n` чисел от 1 до `n` через пробел — содержимое массива после выполнения программы с добавленной строчкой.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
10000	3254 6689 6294 ... 8486 8107
8150 6681 695 ... 5947 4461	

Пояснение к примеру

В примере значение `seed` равно 12345. Полностью пример можно скачать по адресу:

<https://tsweb.ru/trains/shuffle-twice>.

Задача Н. Двухкратная подстрока

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Дана строка S длины n и число k . Найдите в строке S такую подстроку длины k , которая встречается в ней по крайней мере два раза, или выясните, что такой подстроки нет.

Формат входных данных

В первой строке задана строка S ; её длина n — от 1 до 100 000 символов, включительно. Во второй строке задано целое число k ($1 \leq k \leq n$). Строка состоит только из маленьких букв английского алфавита.

Формат выходных данных

Если подстроки длины k , встречающейся хотя бы два раза, не существует, выведите слово «NONE». В противном случае выведите любую из таких подстрок.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
ast 1	NONE
blinkingblueblogger 2	in
aaaaaab 5	aaaaa

Задача I. Плохое хеширование

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Маленький мальчик Петя решает задачу про строки. Не будем останавливаться подробно на её условии. Для нас важно только то, что задача сводится к следующей: дано несколько строк, состоящих исключительно из маленьких букв английского алфавита, и нужно брать пары строк и проверять, равны ли они.

Петя придумал, как делать это быстро: сначала он посчитал *хеш-функцию* от каждой строки, а затем, когда нужно сравнить две строки, просто сравнивает значения хеш-функции от этих строк. Конечно, если значения хеш-функции различны, строки тоже различны. А вот если значения одинаковы, это ещё не гарантирует равенства самих строк.

Мы хотим *взломать* Петино решение, то есть придумать такие две различные строки, что значения хеш-функции от них одинаковы. Чтобы сделать это, разберёмся, что за функцию Петя использует для хеширования.

При ближайшем рассмотрении оказалось, что Петя реализовал *полиномиальный хеш* от строки. Полиномиальный хеш задаётся множителем p и модулем q . Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c . Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}('a') = 97$, $\text{code}('b') = 98$, ..., $\text{code}('z') = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Это достаточно распространённый метод хеширования, однако Петя не подумал о том, что его решение могут взломать, и допустил две существенные ошибки при выборе p и q . Во-первых, модуль q слишком мал, он равен всего лишь 2^{32} (Петя просто считает значение хеш-функции в 32-битном целом беззнаковом типе данных и не обращает внимания на переполнение). Во-вторых, при этом множитель p чётный.

Зная множитель p , взломайте решение Пети.

Формат входных данных

Первая строка ввода содержит целое число p — множитель полиномиального хеширования ($0 < p < 2^{32}$). Гарантируется, что p чётно.

Формат выходных данных

В первых двух строках выведите две различные строки S и T , для которых $h(S) = h(T)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
4	ae ba
1000	vabavydw budqhmng

Пояснения к примерам

В первом примере $h(S) = (97 \cdot 4 + 101) \bmod 2^{32} = 489$ и
 $h(T) = (98 \cdot 4 + 97) \bmod 2^{32} = 489$.

Во втором примере

$h(S) = 118\,097\,098\,097\,118\,121\,100\,119 \bmod 2^{32} = 834\,470\,743$ и
 $h(T) = 98\,117\,100\,113\,104\,109\,110\,103 \bmod 2^{32} = 834\,470\,743$.

Задача J. Взлом хеширования

Имя входного файла: *стандартный ввод*
Имя выходного файла: *стандартный вывод*
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Ваши решения не работают на крайних случаях?
Встроенная быстрая сортировка неожиданно стала работать за квадратичное время?
В геометрических задачах не хватает точности вычислений?
Решение проходит локальное стресс-тестирование, но не работает на тестах жюри?
Именно в вашем случае ошибка оказалась не в решении, а в библиотечной функции?
Хотите узнать, кто за всем этим стоит?
Сегодня у вас есть уникальная возможность вступить в тайную организацию:
Орден Коварных Бобров! Члены этой организации делают в среднем на 146% больше успешных взломов, чем непосвящённые, а в задачи их авторства тесты приходится добавлять в несколько раз реже. Чтобы подать заявку на вступление, необходимо пройти вступительное испытание: решить предложенную ниже задачу.
Торопитесь! Количество мест ограничено!

В этой задаче требуется найти коллизию при полиномиальном хешировании строк, состоящих из маленьких букв английского алфавита.

Полиномиальный хеш строки имеет два параметра: множитель p и модуль q . Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c . Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}('a') = 97$, $\text{code}('b') = 98$, ..., $\text{code}('z') = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

По заданным числам p и q найдите две различные непустые строки A и B такие, что $h(A) = h(B)$.

Формат входных данных

Первая строка ввода содержит два целых числа p и q , разделённых пробелом — параметры функции хеширования ($0 < p < q < 2 \cdot 10^{18}$).

Формат выходных данных

В первых двух строках выведите две различные непустые строки A и B , для которых $h(A) = h(B)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
31 47	aa bq

Пояснение к примеру

В примере $h(A) = (97 \cdot 31 + 97) \bmod 47 = 3104 \bmod 47 = 2$ и $h(B) = (98 \cdot 31 + 113) \bmod 47 = 3151 \bmod 47 = 2$.