1. Гонки

Вася и Петя поссорились и хотят выяснить кто из них прав с помощью забега по очень длинной прямой. Организм Васи устроен так, что он может бежать с постоянной скоростью **v1**(м/с)в течение **k1** секунд, затем ему необходимо отдыхать в течение **r1** секунд, прежде чем он сможет побежать с той же скоростью снова. Аналогично Петя может бежать со скоростью **v2** в течение **k2** секунд и **r2** секунд отдыхать. Вася хочет подготовиться к забегу и просит вас определить кто из двоих мальчиков пробежит большую дистанцию в **n** забегах на различное время **t**(c) (**ti** – время i-го забега).

**Входные данные:**

В первой строке задано натуральное число **n** (n <= 10^6). Следующие **n** строк содержат по 7 целых чисел **v1, k1, r1, v2, k2, r2, ti** (1 <= **v1, k1, v2, k2, ti** <= 10^9 0 <= **r1, r2** <= 10^9).

**Выходные данные:**

Для каждого из t забегов выведите **Vasya**, если первым к финишу придёт Вася, **Petya** – если Петя, и **Draw**, если забег закончится ничьёй.

**Примеры:**

|  |
| --- |
| **входные данные** |
| **3 3 1 3 1 1 2 1 1 1 2 3 1 3 1 2 1 3 1 2 2 2** |
| **выходные данные** |
| **Vasya Petya Draw** |

1. Гипотеза

Давайте познакомимся с гипотезой Коллатца. Проделаем следующий алгоритм с натуральным числом **n**. Если оно чётное, то делим его на 2, а если нечётное, то умножаем на 3 и прибавляем 1 (получаем 3*n* + 1). Над полученным числом выполняем те же самые действия, и так далее. Гипотеза Коллатца заключается в том, что какое бы начальное число *n* мы ни взяли, рано или поздно мы получим единицу.

Доказывать мы её, конечно, не будем, однако проведём некоторые вычисления чтобы убедится в её верности для некоторых чисел.

Дано изначальное число n. Назовём его путём последовательность чисел, которые получаются в ходе преобразования числа n в единицу. Характеристиками пути назовем его длину (количество чисел в последовательности) и максимальное число (встречающееся в последовательности).

Для входного числа **n** определите характеристики его пути.

**Входные данные:**

На вход подаётся натуральное число **n** (n <= 10^TODO).

**Выходные данные:**

Для входного числа **n** выведите длину его пути и максимальное число в нём.

**Примеры:**

|  |
| --- |
| **входные данные** |
| **5** |
| **выходные данные** |
| **6 16** |

Пояснение: путь числа пять следующий – 5 16 8 4 2 1.

1. Кусок строки

Возьмём произвольную строку s, состоящую только из строчных латинских символов. Назовём *полной* такую подстроку, которая содержит все буквы строки и при этом, является строго возрастающей (TODO). Для получения такой подстроки вы используете ваш старый компьютер, который способен только взять какую-либо подстроку фиксированной длины, а затем выполнять только две операции: заменить какой-либо символ подстроки на пустой и поменять местами любой пустой символ подстроки с любым непустым символом из оставшейся строки.

Для данной строки **s** узнайте минимальное количество операций, которые понадобятся чтобы превратить любую подстроку строки **s** в *полную* или выясните, что это невозможно.

**Входные данные:**

На вход подаётся натуральное число **n** – длина строки **s** (n <= 10^5).  
Затем идёт сама строка **s**.

**Выходные данные:**

Выведите одно число - минимальное количество операций, если возможно получить полную подстроку и -1 иначе.

**Примеры:**

|  |
| --- |
| **входные данные** |
| **5 abacb** |
| **выходные данные** |
| **2** |

|  |
| --- |
| **входные данные** |
| **5 bbacb** |
| **выходные данные** |
| **-1** |

Пояснение: в первом примере можем убрать третий символ и поставить на его место четвёртый, тогда получим полную подстроку «abc». Во втором примере ни одну из подстрок не удаётся сделать полной.