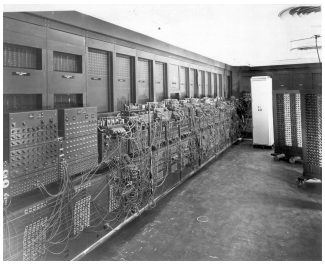
Las claves del ENIAC



Una de las cosas curiosas de viajar al pasado es que tienes al alcance de la mano cosas que en tu tiempo son parte importante de la historia.

De tus lecturas sobre la historia de la informática sabías que el ENIAC fue un ordenador electrónico digital revolucionario en cuya construcción participó, entre otros, von Neumann. Se dice que cuando en 1955 se apagó había hecho más cálculos matemáticos que los realizados por toda la humanidad anteriormente.

Y mira tú por donde, ahora estás en 1955, hace pocos meses de ese apagado, el ENIAC ha sido donado al pueblo de Hill Valley y está en una nave propiedad del ayuntamiento.

Curiosamente la entrada de la sala en la que está la máquina está protegida por una clave de acceso de 5 números que controla... el propio ENIAC.

Los técnicos del ayuntamiento han creado un sistema de claves cambiantes, de forma que cada vez que alguien entra en la sala introduciendo la clave, ésta cambia. Afortunadamente, con unos cuantos sobornos y un poco de ingeniería inversa has llegado a conocer el funcionamiento del algoritmo que va haciendo cambiar las claves y lo has conseguido volcar al CPC-6128 que te trajiste del futuro:

```
// ¡NO hace falta entender este código!
typedef unsigned long long uint64;
uint64 seed;
uint64 state[2] = \{ seed, seed \^\{\} 0x7263d9bd8409f526 \};

uint64 xoroshiro128plus(uint64 s[2]) \{
  uint64 s0 = s[0];
  uint64 s1 = s[1];
  uint64 result = s0 + s1;
  s1 \^\{\}= s0;
  s[0] = ((s0 $<$$<$ 55) | (s0 $>$$>$ 9)) \^\{\} s1 \^\{\} (s1 $<$$<$ 14);
  s[1] = (s1 $<$$<$ 36) | (s1 $>$$>$ 28);
  return result;
\}

void initSystem(int seed) \{
  state[0] = seed;
  state[1] = ((uint64)seed) \^\{\} 0x7263d9bd8409f526;
\}

int clave() \{
  return xoroshiro128plus(state) \% 100000;
\}
```

Cada mañana, el funcionario de turno ejecuta la función `initSystem` con una *semilla* concreta. Después, cada vez que alguien mete un número, comprueba si éste es igual al que devuelve la función `clave()` (que cada vez devuelve una cosa).

Como ejemplo, si un día se inicia el sistema con `initSystem(123)`, las primeras claves serán:`31256`, `38526`, `87793`, `49812`, `71504`, `53872`, `4234`, `44609` y `85102`.

Afortunadamente, has encontrado también una vulnerabilidad en el sistema: si la clave *no* tiene ningún 7, la puerta se abre aunque no aciertes.

Antes de colarte con tus padres en el ayuntamiento para ver el ENIAC, quieres saber cuántos intentos como mucho tendrás que hacer (metiendo cualquier cifra) antes de que se abra la puerta por el fallo anterior. O lo que es lo mismo, cuál es la secuencia de claves más larga en la que ninguna de las claves tienen sietes.

Entrada

La entrada estará compuesta por distintos casos de prueba, cada uno ocupando una única línea.

Cada caso tiene dos números: s y n. El número s es el valor que el funcionario ha dado en el momento de inicializar el sistema llamando a `initSystem`. El segundo es el número de claves totales en el que buscar la secuencia más larga (hasta 500.000).

La entrada termina con un caso con dos ceros que no debe procesarse.

Salida

Se escribirá un único número con la longitud de la secuencia más larga de claves válidas que *no* tienen ningún 7.

Entrada de ejemplo

```
123 1
123 2
123 5
123 9
0 0
```

Salida de ejemplo

```
1
2
2
3
```

Notas

Ten en cuenta que:

- Para que el ejercicio sea evaluado debe ser primero aceptado por el juez.
- Debes tener al menos las dos funciones siguientes:
  1. `bool tieneSietes(int n)` que devuelve `true` si el entero n no necesita ningún 7 para escribirse en base 10.
  2. `int secuenciaMasLarga(int n)` que llama n veces a `clave()` y devuelve la longitud de la secuencia más larga de valores devueltos que no tenían ningún 7. Esta función *no* debe necesitar espacio adicional en forma de vector.
- El ejemplo de entrada se corresponde con el ejemplo que aparece en el enunciado, inicializando con `123`. Según vemos en la secuencia de claves, hay un primer bloque de dos claves sin 7 que hace que varias de las consultas respondan con un 2. Tenemos que esperar hasta la novena clave para ver una secuencia de tres claves seguidas sin 7, lo que se refleja en el último caso de prueba.

Nota

Este ejercicio debe verse en el contexto de la asignatura de Fundamentos de Algoritmia (FAL), FDI-UCM 2019/2020 (prof. Marco Antonio Gómez Martín). Por tanto *no* vale cualquier solución, sino sólo aquellas que utilicen los conceptos de FAL. Es muy posible que se den aclaraciones adicionales en clase a este respecto.