Computando patentes

Contribución de Agustín Santiago Gutiérrez (basado en sugerencia de Pablo Heiber)

Descripción del problema

En Argentina conviven actualmente dos sistemas principales para la numeración de las matrículas automovilísticas, también denominadas *patentes*:

1. Un formato utilizando desde 1995 hasta 2016, en el cual la patente de un vehículo es una cadena de **exactamente seis caracteres**. Los primeros 3 de estos caracteres son letras **mayúsculas** y los últimos 3 son dígitos. Algunos ejemplos: BET123, PAT180, BAD666, AAA000 y ZZZ381
2. Un formato utilizado desde 2016 hasta hoy, en el cual la patente de un vehículo es una cadena de **exactamente siete caracteres**. Los primeros 2 y los últimos 2 son letras **mayúsculas** y los 3 del medio son dígitos. Algunos ejemplos: AB123CD, AG759LH, ZZ100XX

En ambos formatos, se entiende por *letra* a cualquiera de los 26 caracteres del alfabeto inglés (es decir, el alfabeto castellano sin la Ñ, desde A hasta Z) y por *dígito* a uno de los 10 posibles caracteres numéricos desde 0 hasta 9 inclusive.

En los dos casos, la patente *siguiente* de una dada es la que se obtiene **avanzando** el último caracter, es decir, **reemplazándolo por el caracter siguiente correspondiente**. Por ejemplo la patente siguiente de ABC151 es ABC152, y la patente siguiente de TE200AW es TE200AX.

Cuando el último caracter es 9 o Z, que no tienen un caracter siguiente, se vuelve a comenzar por 0 o por A, y además se pasa a **avanzar el anteúltimo caracter de la patente**. Si este también era 9 o Z, este proceso continúa hasta conseguir la siguiente patente. Por ejemplo, la patente siguiente de AB999ZZ es AC000AA, y la patente siguiente de DOW199 es DOW200.

Para ayudar a la *Dirección Nacional de los Registros Nacionales de la Propiedad del Automotor y de Créditos Prendarios*, debes escribir un programa que dada una patente, la cual puede estar en cualquiera de los dos formatos, calcule la K-ésima patente siguiente en el mismo formato. Es decir, si $K = 1$ se debe calcular la patente siguiente, como en los ejemplos ya vistos. Si $K = 2$ se debe tomar la siguiente de la siguiente, y así.

Detalles de implementación

Debes implementar la función `siguiente(patente, K)`, que recibe una cadena de caracteres `patente`, que contiene la patente en alguno de los dos formatos, y el entero K.

La función debe retornar una cadena de caracteres que contenga la K-ésima siguiente patente.

Se garantiza que la patente recibida es válida en alguno de los dos formatos, y que existe una K-ésima siguiente patente. Por ejemplo, si $K = 1$, se garantiza que no se recibirá la patente ZZZ999 ni la ZZ999ZZ.

Evaluador local

El evaluador local lee de la entrada estándar:

- Una línea con la patente
- Una línea con el valor K

Escribe a la salida estándar una línea con el resultado de llamar a la función `siguiente` con la patente y el valor K recibidos por la entrada estándar.

Ejemplos

Si el evaluador local recibe la siguiente entrada:

```
ABC151
1
```

Una implementación correcta deberá devolver:

```
ABC152
```

Si en cambio recibe:

```
TE200AW
5
```

Para una implementación correcta devolverá:

```
TE200BB
```

Subtareas

- Para $K = 1$ (Total 50 ptos.):
  1. Patentes terminadas en 0 (5 puntos)
  2. Patentes que **no** terminan ni en 9, ni en Z (7 puntos)
  3. Patentes de formato 1 (17 puntos)
  4. Patentes de formato 2 (21 puntos)
- Para $1 \leq K \leq 100$ (Total 40 ptos.):
  5. Patentes de formato 1 (18 puntos)
  6. Patentes de formato 2 (22 puntos)
- Para $1 \leq K < 456.976.000$ (Total 10 ptos.):
  7. Patentes de formato 1 (4 puntos)
  8. Patentes de formato 2 (6 puntos)