Carnet por puntos 2

La DGT nos ha pedido ayuda para gestionar el *carnet por puntos*. Los conductores están identificados de manera unívoca por su DNI y la cantidad de puntos de un conductor está entre 0 y 15 puntos inclusives.

La implementación del sistema se deberá realizar como un TAD `carnet_puntos` con las siguientes operaciones:

- `nuevo(dni)`: añade un nuevo conductor identificado por su `dni` (un `string`), con 15 puntos. En caso de que el DNI esté duplicado, la operación lanza una excepción `domain_error` con mensaje `Conductor duplicado`.
- `quitar(dni, puntos)`: le resta puntos a un conductor tras una infracción. Si a un conductor se le quitan más puntos de los que tiene, se quedará con 0 puntos. En caso de que el conductor no exista, lanza una excepción `domain_error` con mensaje `Conductor inexistente`.
- `recuperar(dni, puntos)`: le añade puntos a un conductor enmendado. Si debido a una recuperación un conductor supera los 15 puntos, se quedará con 15 puntos. En caso de que el conductor no exista, lanza una excepción `domain_error` con mensaje `Conductor inexistente`.
- `consultar(dni)`: devuelve los puntos actuales de un conductor. En caso de que el conductor no exista, lanza una excepción `domain_error` con mensaje `Conductor inexistente`.
- `cuantos_con_puntos(puntos)`: devuelve cuántos conductores tienen un determinado número de puntos. En caso de que el número de puntos no esté entre 0 y 15 lanza una excepción `domain_error` con mensaje `Puntos no validos`.
- `lista_por_puntos(puntos)`: produce una lista con los DNI de los conductores que poseen un número determinado de puntos. La lista estará ordenada por el momento en el que el conductor pasó a tener esos puntos, primero el que menos tiempo lleva con esos puntos. En caso de que el número de puntos no esté entre 0 y 15 lanza una excepción `domain_error` con mensaje `Puntos no validos`.

Requisitos de implementación.

La implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD, implementar las operaciones y justificar la complejidad resultante.

Los métodos del TAD no deben mostrar nada por pantalla. El manejo de la entrada y salida de datos se realizará en funciones externas al TAD.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso está formado por una serie de líneas, en las que se muestran las operaciones a llevar a cabo, una por cada línea: el nombre de la operación seguido de sus argumentos. La palabra `FIN` en una línea indica el final de cada caso.

Salida

Para cada caso de prueba se escribirán los datos que se piden. Las operaciones que generan salida son:

- `consultar`, que debe escribir una línea con `Puntos de DNI: N`, donde `DNI` es del DNI por el que se ha preguntado y `N` sus puntos;
- `cuantos_con_puntos`, que debe escribir una línea con `Con N puntos hay M`, donde `N` son los puntos por los que se ha preguntado y `M` cuántos tienen esos puntos;
- `lista_por_puntos`, que escribe una línea con `Tienen N puntos:`, seguida de los DNIs de los que tienen esos puntos, separados por espacios en blanco.

Cada caso termina con una línea con tres guiones (`---`).

Si una operación produce un error, entonces se escribirá una línea con `ERROR:`, seguido del error que devuelve la operación, y no se escribirá nada más para esa operación.

Entrada de ejemplo

```
nuevo 123A
nuevo 456B
nuevo 666
cuantos_con_puntos 15
cuantos_con_puntos 0
lista_por_puntos 15
quitar 666 15
lista_por_puntos 0
quitar 456B 9
consultar 456B
quitar 123A 10
recuperar 123A 1
lista_por_puntos 6
recuperar 123A 1
lista_por_puntos 6
lista_por_puntos 7
FIN
nuevo 123A
nuevo 123A
cuantos_con_puntos 20
quitar 456B 2
FIN
```

Salida de ejemplo

```
Con 15 puntos hay 3
Con 0 puntos hay 0
Tienen 15 puntos: 666 456B 123A
Tienen 0 puntos: 666
Puntos de 456B: 6
Tienen 6 puntos: 123A 456B
Tienen 6 puntos: 456B
Tienen 7 puntos: 123A
---
ERROR: Conductor duplicado
ERROR: Puntos no validos
ERROR: Conductor inexistente
---
```

Autor: Alberto Verdejo.