Carnet por puntos

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

- `crea()`: Crea un sistema de gestión de carnets por puntos vacío.
- `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 un error (excepción `EConductorDuplicado`).
- `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 un error (excepción `EConductorNoExiste`).
- `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 un error (excepción `EConductorNoExiste`)
- `consultar(dni)`: Devuelve los puntos actuales de un conductor. En caso de que el conductor no exista, lanza un error (excepción `EConductorNoExiste`).
- `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 un error (excepción `EPuntosNoValidos`).
- `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 un error (excepción `EPuntosNoValidos`).

Trabajo a realizar

Se debe completar un programa que lea y ejecute una serie de acciones a realizar sobre un sistema de gestión de carnet por puntos. Las posibles órdenes son las siguientes:

- `nuevo` <DNI>: Da de alta un nuevo conductor en el sistema. Imprime OK si el alta tiene éxito, o `CONDUCTOR_YA_EXISTE` en caso de que trate de darse de alta un conductor que ya existe.
- `quitar` <DNI> <PUNTOS>: Quita puntos a un conductor. Imprime el número de puntos del conductor tras la operación, o `CONDUCTOR_NO_EXISTE` en caso de que el conductor no esté dado de alta en el sistema.
- `recuperar` <DNI> <PUNTOS>: Permite que un conductor recupere puntos. Imprime el número de puntos del conductor tras la operación, o `CONDUCTOR_NO_EXISTE` en caso de que el conductor no esté dado de alta en el sistema.
- `consultar` <DNI>: Consulta el número de puntos de un conductor. Imprime dicho número, o `CONDUCTOR_NO_EXISTE` en caso de que el conductor no esté dado de alta en el sistema.
- `cuantos_con_puntos` <PUNTOS>: Imprime el número de conductores que tienen el número de puntos indicado. Si PUNTOS no está entre 0 y 15, imprime `PUNTOS_NO_VALIDOS`.
- `lista_por_puntos` <PUNTOS>: Imprime los DNIs de los conductores que tienen los puntos indicados, ordenados por el momento de obtención de dichos puntos (primero los que hace menos tiempo que los han obtenido). Si `PUNTOS` no está entre 0 y 15, imprime `PUNTOS_NO_VALIDOS`.

Ejemplo de entrada / salida

Entrada	Salida
nuevo 1234	OK
nuevo 1234	CONDUCTOR_YA_EXISTE
consultar 1234	15
consultar 1235	CONDUCTOR_NO_EXISTE
nuevo 1235	OK
consultar 1235	15
cuantos_con_puntos 15	2
cuantos_con_puntos 8	0
cuantos_con_puntos 16	PUNTOS_NO_VALIDOS
cuantos_con_puntos 15	2

cuantos_con_puntos 14	0
lista_por_puntos 15	1235 1234
lista_por_puntos 16	PUNTOS_NO_VALIDOS
recuperar 1234 1	15
cuantos_con_puntos 15	2
lista_por_puntos 15	1235 1234
quitar 1234 1	14
cuantos_con_puntos 15	1
cuantos_con_puntos 14	1
lista_por_puntos 15	1235
lista_por_puntos 14	1234
recuperar 1234 10	15
lista_por_puntos 15	1234 1235
recuperar 1456 5	CONDUCTOR_NO_EXISTE
quitar 1456 5	CONDUCTOR_NO_EXISTE

Se proporciona el archivo `main.cpp` en el que se implementa la lógica de entrada/salida necesaria. El código proporcionado no debe modificarse.

Se proporciona, así mismo, el archivo `CarnetPorPuntos.h`, con la definición de las operaciones de TAD. Dicho archivo debe completarse eligiendo una representación apropiada para gestionar la información del sistema. Pueden, asimismo, definirse todos los métodos privados adicionales en `CarnetPorPuntos` que se estimen oportunos.

Las operaciones del TAD deben implementarse en el archivo `CarnetPorPuntos.cpp` que también se proporciona. Debe, asimismo, determinar justificadamente la complejidad de cada operación.