Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Segundo Cuatrimestre, 12 de septiembre de 2017

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. (2.5 puntos) Extiende el TAD `Cola` implementado con lista enlazada simple visto en clase con una nueva operación cuya cabecera en `C++` es

```
void llevarAlPrincipio(unsigned int pos);
```

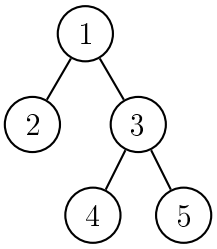
que lleva el elemento que está en la posición *pos* de la cola a la primera posición de la misma. Ten en cuenta que $pos = 1$ es el primer elemento de la cola; $pos = 2$ es el segundo; y así sucesivamente.

Si la posición de la cola no existe, deberá señalarse un error "Posicion inexistente".

Aparte de implementar esta operación, debes indicar la complejidad de la misma.

Para resolver este ejercicio no se puede crear ni destruir memoria dinámica, ni tampoco modificar los valores almacenados en la cola.

2. (2.5 puntos) Implementa una función que, dado un árbol binario de enteros y un número entero no negativo k, determine el número de hojas cuya profundidad es mayor que k. Por ejemplo, para el siguiente árbol



la función devolvería 3 si $k = 0$ o $k = 1$, devolvería 2 si $k = 2$ y devolvería 0 si $k \geq 3$.

Aparte de implementar este subprograma, debes indicar la complejidad del mismo.

3. (5 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 inclusives. La implementación del sistema se deberá realizar como un TAD `CarnetPorPuntos` 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 un error "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. Si los puntos resultantes de esta operación son los mismos que los que tiene el conductor actualmente, entonces la operación debe ignorarse. En caso de que el conductor no exista, lanza un error "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. Si los puntos resultantes de esta operación son los mismos que los que tiene el conductor actualmente, entonces la operación debe ignorarse. En caso de que el conductor no exista, lanza un error "Conductor inexistente".
- `consultar(dni)`: Devuelve los puntos actuales de un conductor. En caso de que el conductor no exista, lanza un error "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 un error "Puntos no válidos".
- `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 "Puntos no válidos".

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.