Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Segundo Cuatrimestre, 27 de Junio de 2017.

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 desplaza(unsigned int pos, unsigned int dist);
```

que desplaza al elemento que está en la posición `pos` de la cola un número `dist` de posiciones hacia la izquierda. Ten en cuenta que $pos = 1$ es el primer elemento de la cola; $pos = 2$ es el segundo; y así sucesivamente.

Si el valor de `dist` es demasiado grande y no hay suficientes posiciones para hacer el desplazamiento, el elemento quedará situado en la primera posición de la cola.

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. (3 puntos) Un nodo de un árbol binario de enteros se dice que es *singular* si la suma de los valores almacenados en sus nodos antepasados es igual a la suma de los valores almacenados en sus nodos descendientes. Implementa un subprograma que, dado un árbol binario de enteros, devuelva el número de nodos singulares que tiene.

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

3. (4.5 puntos) Nos han encargado implementar un sistema para la gestión de la admisión en el Servicio de Urgencias de un Hospital. Cuando un paciente llega al servicio, se le toman los datos, se le asigna un código de identificación único (un número entero no negativo), y también se le asigna un nivel de urgencia, dependiendo de su estado (leve, normal, o grave). A partir de ahí, el paciente espera a ser atendido. El orden de atención da prioridad a los pacientes graves sobre los normales, y a los normales sobre los leves. Una vez atendido un paciente, sus datos se eliminan del sistema. Así mismo, en cualquier momento un paciente puede desistir de ser atendido. En este caso, en el control de salida se le solicita su número de identificación, y se elimina todo rastro de él del sistema.

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

- `crea()`: Operación constructora que crea un sistema de gestión de admisiones vacío.
- `an_paciente(codigo, nombre, edad, sintomas, gravedad)`: Añade al sistema un nuevo paciente con código de identificación `codigo`, con nombre `nombre`, con edad `edad`, con una descripción de síntomas `sintomas`, y con código de gravedad `gravedad`. En caso de que el código esté duplicado, la operación señala un error "Paciente duplicado".
- `info_paciente(codigo, nombre, edad, sintomas)`: Almacena en `nombre`, `edad` y `sintomas` la información correspondiente del paciente con código `codigo`. En caso de que el código no exista, levanta un error "Paciente inexistente".
- `siguiente(codigo, gravedad)`: Almacena en `codigo` y `gravedad`, respectivamente, el código y la gravedad del siguiente paciente a ser atendido. Como se ha indicado antes, se atiende primero a los pacientes graves, después a los de nivel de gravedad normal, y por último a los leves. Dentro de cada nivel, los pacientes se atienden por orden de llegada. En caso de que no haya más pacientes, esta operación levanta un error "No hay pacientes".
- `hay_pacientes()`. Devuelve `true` si hay más pacientes en espera, y `false` en otro caso.
- `elimina(codigo)`: Elimina del sistema todo el rastro del paciente con código `codigo`. Si no existe tal paciente, la operación no tiene efecto.

Dado que este es un sistema crítico, 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.