Programación II. Parcial 1. 14 de Marzo 2017

Apellidos: _____Nombre:_____

Ejercicio 1. (3 puntos) Una empresa aseguradora de coches va a reformar su software de gestión interna, y para ello nos ha pedido que diseñemos los nuevos TAD que van a implementar la gestión de la información interna. Para ello, tenemos que tener en cuenta que:

- La empresa cuenta con un conjunto de clientes, que se supone de un máximo de un millón. De cada cliente tenemos la siguiente información: nombre (máximo 80 caracteres), id interno (es un long int), y los vehículos que tiene asegurados. Asumiremos que cada cliente puede tener 10 vehículos como máximo.
- Cada vehículo asegurado está descrito por la matrícula y el id del cliente asociado. Se asume que el total de vehículos asegurados no es mayor que dos millones.
- La empresa cuenta también con pólizas de hogar, cada una de ellas descrita por el id de la póliza y la dirección de la casa (cadena de como mucho 100 caracteres). Cada cliente va a tener asociado como mucho una póliza de hogar.

a) **(1 punto)** ¿Cuántos Tipos Abstractos de Datos necesitas para representar lo descrito anteriormente, y qué nombres les darías? Los tipos Status y Boolean se encuentran ya definidos (no se consideran aquí)

Nº TADs: ___ Nombres TADs:

_____

Escribe el código en C de las Estructuras de Datos (EdD) necesarias para implementar estos TADs y di en qué ficheros estarían esos códigos.

b) **(1 punto)** Proporciona el código C con control de errores de la primitiva altaCliente, que toma el nombre de un nuevo cliente y lo da de alta comprobando si anteriormente no existe:

Status altaCliente(Clientes *pcs, const char *nombre)

¿En qué TAD estaría implementada esta primitiva?

¿Qué primitivas necesitas usar para su implementación y a qué TADs pertenecen? No hace falta que implementes estas últimas, sólo dar sus prototipos.

c) **(1 punto)** Proporciona el código C con control de errores de la primitiva:

Status imprimeVehiculosCliente (const Aseguradora *pa, const char *nombre, FILE *f)

Que dados los datos de la aseguradora, y el nombre de un conductor, imprima en el fichero que proporcionamos los datos de todos los vehículos que tiene asegurados en la empresa.

¿En qué TAD / archivo estará definido los detalles del código de dicha función? ¿hace uso de otras primitivas de otros TAD? ¿de cuáles? Y a su vez, ¿estas hacen uso de otras primitivas? ¿de cuáles?

Programación II. Examen parcial de evaluación continua. Marzo 2017

Apellidos: _____ Nombre: _____ Grupo: ____

Ejercicio 2 (2 puntos). Traduce la siguiente expresión a sufijo (posfijo) utilizando el algoritmo de traducción estudiado en clase:

A / (B – C + D * F) - G ;

Dibuja la evolución del algoritmo sobre la expresión paso a paso:

a) Traduce la siguiente expresión a prefijo, aplicando el algoritmo correspondiente y mostrando su evolución paso a paso:

A B C D - E * F G + / * + ;

Programación II. Examen parcial de evaluación continua. Marzo 2017

Apellidos: _____ Nombre: _____ Grupo: ____

Ejercicio 3 (2.5 puntos). Se desea crear un editor de texto científico que utiliza marcas para dar formato a fórmulas matemáticas.

En este editor el texto se compone de secuencias de caracteres explícitamente identificados y delimitados por marcas, pudiendo cada una de las secuencias de caracteres estar compuestas por otras secuencias. Por ejemplo, una fórmula matemática estaría delimitada por la marca de apertura `\begin{eqn}` y finalizaría con la marca de cierre `\end{eqn}` mientras que una fracción dentro de esa fórmula estaría delimitada por las marcas `\begin{frac}` y finalizaría con la marca de cierre `\end{frac}`

Es decir, cada una de las partes que componen una fórmula se delimita mediante dos etiquetas, una de apertura/inicio de elemento `\begin{tag}` y otra de cierre/fin `\end{tag}`, donde `tag` se debe sustituir por el nombre de etiqueta que se elija, p.e. `\begin{eqn}` y `\end{eqn}`, o `\begin{frac}` y `\end{frac}` Así, entre una etiqueta de apertura y la correspondiente de cierre puede haber o bien otro(s) par(es) de etiquetas diferentes anidadas o bien valores literales, por ejemplo la fórmula $\frac{3x}{5y}$ se escribiría en el editor como:

```
\begin{eqn}
    \begin{frac}
        \begin{num} 3 x  \end{num}
        \begin{den} 5 y  \end{den}
    \end{frac}
\end{eqn}
```

Asúmase que existe una rutina que procesa el fichero como una cadena de "tokens" (etiquetas o valores) extrayendo iterativamente token a token. Para el ejemplo anterior: `\begin{eqn}`, `\begin{frac}`, `\begin{num}`, `3`, `x`, `\end{num}`, `etc`.

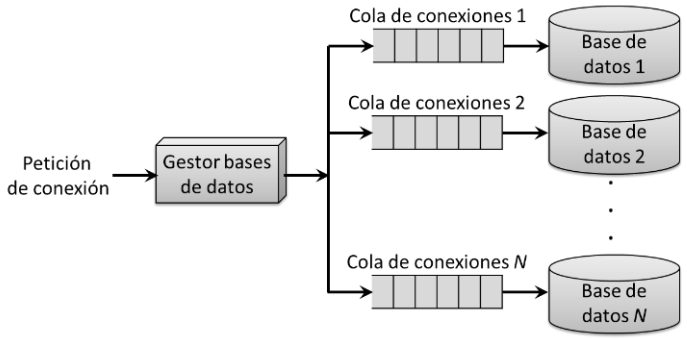
Usando esa rutina, se desea desarrollar un programa que valide la sintaxis de un fichero comprobando la corrección de las etiquetas de apertura y cierre, y teniendo en cuenta la anidación de pares de etiquetas.

a) ¿Qué TAD y algoritmo de los estudiados en la asignatura se podrían aplicar para tal programa? Razona brevemente la respuesta.
b) Indica gráficamente, token a token, el estado del TAD y la evolución del algoritmo anterior sobre el fichero de arriba.
a) Da el pseudocódigo del algoritmo anterior.

Programación II. Examen parcial de evaluación continua. Marzo 2017

Apellidos: _____ Nombre: _____ Grupo: ____

Ejercicio 4 (2.5 puntos). Un **gestor de bases de datos** es un programa que controla el almacenamiento, modificación y extracción de información de un conjunto de bases de datos mediante conexiones a las mimas. Como se muestra en la figura de abajo, una situación típica es que cada base de datos tiene una cola donde almacena las peticiones de conexión que le llegan para procesarlas según va teniendo disponibilidad.



Suponer que se dispone de las siguientes estructuras de datos para almacenar los datos de un gestor y sus colas.

```
typedef struct {
    Cola *colaConexiones;
    char *nombre;
} BaseDatos;
typedef struct {
    BaseDatos *basesDatos;
    int numBasesDatos;
} GestorBasesDatos;
```

Al llegar una petición de conexión al gestor, éste comprobará la cola de la base de datos solicitada, que está identificada por un "nombre". En caso de que la cola de esa base de datos no esté llena, la petición se almacena en esa cola. En caso contrario, se rechaza la petición.

Con todo lo anterior, se pide implementar la función:

```
status gestorBaseDatos_solicitarConexion(GestorBasesDatos *gestor, Conexion *conexion);
```

que inserta la conexión de entrada en la cola de la base de datos correspondiente.

Asumir que existe la siguiente función para obtener una cadena de caracteres con una copia del nombre de la base de datos de una conexión:

```
char *conexion_getNombreBaseDatos(Conexion *);
```

La función ha de tener los siguientes valores de retorno:

- `OK`, si la petición se pudo insertar en la cola de la base de datos
- `ERROR_ARGUMENTOS`, si alguno de los argumentos de entrada es incorrecto
- `ERROR_SOBRECARGA`, si la cola de la base de datos está llena
- `ERROR_INTERNO`, si se produjo algún error durante la ejecución

Asumir que existen las primitivas de los TAD Cola y Elemento vistas en clase:

```
Cola *cola_crear();
void cola_liberar(Cola *pq);
```

(sigue por detrás):

```
Bool cola_vacia(const Cola *pq);
Bool cola_llena(const Cola *pq);
Status cola_insertar(Cola *pq, const Elemento *pe);
Elemento *cola_extraer(Cola *pq);
void elemento_liberar(Elemento *);
Elemento *elemento_copiar(const Elemento *);
```

y la siguiente función de creación de un elemento de cola a partir de una petición:

```
Elemento *elemento_crear(Conexion *);
```