Algoritmos y Estructura de Datos. K1025. 2do Examen Parcial.
Fecha: 25/10/2017
Apellido y nombre: …… **Legajo:** ……..

Para aprobar debe sumar como mínimo 60 puntos (calif. 6), siendo 80 puntos el mínimo requerido para aprobación directa (calificación 8).

1) Algoritmo. Seleccione la respuesta correcta e ingrese un comentario describiendo su decisión.

```
template <typename T, typename K> int miFuncion(FILE*
f, K v, int (*criterio)(T,K)) {
        int i = 0;
        int j = fileSize<T>(f)-1;
        int k = (i+j)/2;
        seek<T>(f,k);
        T r = read<T>(f);
        while( i<=j && criterio(r,v)!=0 ) {
                if( criterio(r,v)>0 ) {
                        j = k-1;
                } else {
                        if( criterio(r,v)<0 ) {
                                i=k+1;
                        }
                }
                k = (i+j)/2;
                seek<T>(f,k);
                r = read<T>(f);
        }
        return i<=j?k:-1;
}
```

¿Qué resultado produce esta función?

[ ] Inserta registros ordenados por criterio

[ ] Elimina registros que superen cierto tamaño

[ ] Retorna la posición del registro que coincide con v

[ ] Ninguna de las 3 anteriores

Comentario/supuestos sobre la respuesta seleccionada:

……

……

……

(20 puntos)

2) HTML. Se precisa validar la sintaxis del código fuente HTML. Para ello se debe analizar el equilibrio de las etiquetas. El programa debe evaluar las palabras clave de apertura, es decir: <body>, o <h1> y, cuando encuentra una de cierre, evaluar si coincide con la correspondiente, o sea: </h1>, o </body>, por ejemplo:

Correcto: <body> <h1> </h1> </body>
Correcto: <body> <h1> Texto del encabezado </h1> </body>
Incorrecto: <body> </h1> </body>
Incorrecto: <body> <h1>   </body>

Se pide: Crear la o las estructuras necesarias y la función ***evaluar()*** que recibirá como parámetro un array de cadenas de caracteres con todas las palabras del archivo html e iterarlo evaluando si existe alguna inconsistencia. La función debe retornar true (1) si es válido o false (0) en caso contrario.

Info: Para comparar cadenas puede usar la función compare() de strings, que retorna 0 cuando coinciden, por ejemplo:

```
string curso = “k1025”;
if(curso.compare("k1025") == 0) cout << “son iguales”;
```

(30 puntos)

3) Streaming. Se debe crear una función que cargue en memoria las canciones más solicitadas desde un archivo binario de registros, de modo tal de agilizar el tiempo de respuesta del servicio (caché). El archivo posee las canciones que superaron las 10000 reproducciones del día anterior, por lo que la cantidad siempre es variable. Cada registro de canción posee la siguiente estructura:

id_cancion	**id_album**	**orden**	**audio**
entero	entero	entero	cadena de caracteres [2048]

Donde el campo **audio** contiene la canción codificada en Base64. La función debe organizar en una estructura de datos a todos los registros leídos del archivo, agrupándolos por álbum y en la secuencia indicada por el valor del campo **orden**.

Se pide: Crear la función ***cargarCache()*** que recibe por parámetro la ruta al archivo y deberá: 1) leer y agrupar todas las canciones por **id_album**, 2) retornar una lista de álbumes con canciones enlazadas ascendentemente (0-9) por el valor del campo **orden**. Tener presente que cada álbum puede tener más de una canción, pero no puede haber repeticiones de un mismo álbum en la estructura. Crear las estructuras necesarias.
(50 puntos)