

**Algoritmos y Estructura de Datos. K1041. 2º Examen Parcial.**

**Fecha: 27/10/2017**

**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.

<pre> template &lt;typename T&gt; Nodo&lt;T&gt;* miFuncion(Nodo&lt;T&gt;* la, Nodo&lt;T&gt;* lb, Nodo&lt;T&gt;* lc) {     while(la!=NULL&amp;&amp;lb!=NULL) {         if(la-&gt;info&lt;=lb-&gt;info) {             agregar(lc,la-&gt;info);             la=la-&gt;sig;         } else {             agregar(lc,lb-&gt;info);             lb=lb-&gt;sig;         }     }     while(la!=NULL) {         agregar(lc,la-&gt;info);         la = la-&gt;sig;     }     while(lb!=NULL) {         agregar(lc,lb-&gt;info);         lb = lb-&gt;sig;     }     return lc; } </pre>	<p>¿Qué resultado produce esta función?</p> <p><input type="checkbox"/> Inserta nodos ordenados por el campo info</p> <p><input type="checkbox"/> Intercala los nodos de dos listas</p> <p><input type="checkbox"/> Retorna el nodo que coincide con T</p> <p><input type="checkbox"/> Ninguna de las 3 anteriores</p> <p>Comentario/supuestos sobre la respuesta seleccionada:</p> <p>.....</p> <p>.....</p> <p>.....</p>
--	--

(20 puntos)

**2) Chat.** Un sistema de atención al público necesita poner en espera las solicitudes de conversación con los representantes. Como la demanda supera a la cantidad de operadores, el objetivo es que los clientes queden en espera manteniendo el orden de llegada. Cada cliente es representado en el sistema con los siguientes datos:

id_cliente	nickname
entero	cadena de caracteres

Se pide: Crear un programa que agregue clientes a una estructura de datos desde la función **agregarCliente()** a medida que se van registrando en el sistema.

También implementar la función **atenderProximo()** que retorne los datos del siguiente cliente a atender. Crear las estructuras necesarias.

(30 puntos)

**3) Videojuego.** Se debe crear una función que cargue en memoria el historial de partidos de fútbol disputado por cada equipo desde un archivo binario de registros. El videojuego agrupa los datos históricos en un array de hasta 255 equipos, donde se espera que cada elemento del array permita acceder al conjunto de registros históricos de cada equipo ordenados por el campo **fecha**. El archivo posee registros con los datos de todos los partidos jugados por cada equipo. Cada registro de partido posee la siguiente estructura:

id_equipo	id_partido	fecha	puntos	jugadores
entero (0-254)	entero	entero largo	entero	array de enteros [18]

El campo **fecha** contiene el resultado de llamar a la función `time(NULL)` de la biblioteca `time.h`, por lo tanto es un valor entero desde donde es posible obtener la secuencia de partidos.

Se pide: Crear la función **cargarHistorico()** que recibe por parámetro, el array (con todos los elementos inicializados en NULL), la ruta al archivo y deberá: 1) leer y agrupar todos los registros de partidos por **id\_equipo**, 2) enlazar todos los partidos ascendentemente (0-9) según el valor del campo **fecha**, siendo que los registros pueden haber sido guardados en el archivo sin contemplar ningún tipo de orden. Crear las estructuras necesarias.

(50 puntos)