UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Algoritmos y Estructuras de Datos. K1041. 1er Recuperatorio 2da Evaluación Parcial.
Fecha: 22/11/2019
Apellido y nombre: …… **Legajo:** ……..

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

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

```
void miFuncion(Nodo*& o) {
  Nodo* pre = o->ant;
  Nodo* pro = o->sig;
  pre->sig = pro;
  pro->ant = pre;
  delete o;
}
```

¿Qué resultado produce esta función?

[ ] Elimina la estructura

[ ] Reemplaza un nodo de la estructura

[ ] Ordena los nodos según **o**

[ ] Ninguna de las anteriores

Comentario/supuestos sobre la respuesta seleccionada:

……..……..……..……..……..……

……..……..……..……..……..……..

(20 puntos)

2) PBX. El software de una centra telefónica necesita incorporar una nueva funcionalidad. Se desea implementar que la central pueda atender todas las llamadas entrantes y queden en espera hasta que un operador tome la llamada. Cada llamada posee los siguientes datos:

idLlamada (entero)	numeroOrigen (entero)	idLinea (entero)	fechaHoraIngreso (entero largo)	codigoCancionEspera (cadena de caracteres)

Se debe cargar en la memoria temporal (buffer) de la central a cada nueva llamada, para que luego se atiendan manteniendo el orden de llegada.

Se pide: Crear la función ***nuevaLlamada()*** que recibe por parámetro la estructura en memoria y una variable con la nueva Llamada a poner en espera. El atributo **fechaHoraIngreso** se debe cargar con el resultado de invocar a la función `time(NULL)` al momento de poner en espera. También desarrollar la función ***atenderSiguiente()***, que recibe por parámetro (y por referencia): el buffer, una variable donde se acumulan los milisegundos de espera, otra con la cantidad de llamadas atendidas e invocará a la función provista ***atender()*** pasando por parámetro la próxima Llamada. También, actualizar las variables acumuladoras de milisegundos y cantidad de llamadas. La función retornará -1 si no encontró llamadas en espera y 0 en caso contrario. Crear las estructuras necesarias.

(30 puntos)

3) Servidores. Una plataforma de videojuegos on-line mantiene en memoria todos los servidores de todos los videojuegos disponibles en tiempo real. La plataforma puede gestionar hasta 128 títulos (juegos) diferentes y por cada uno mantiene un listado de servidores disponibles. Cada registro de servidor posee los siguientes datos:

idJuego (entero 0-127)	nombreJuego (array de car. [15])	idServidor (entero)	nombreServidor (array de car. [15])	ping (entero)	cantJugadores (entero)

donde **ping** es la latencia actual del servidor en milisegundos.
Se pide: Cargar la estructura de datos con los registros de servidores a leer desde un archivo invocando a la función ***cargarServidores()***, que recibirá por parámetro la estructura, la ruta al archivo y deberá agregar los servidores leídos ordenados ascendentemente por el campo **ping**. Si la estructura ya tiene ese servidor, deberá reemplazarlo. También crear la función ***buscar()***, que recibirá por parámetro la estructura, un idJuego, un idServidor y retornará un puntero al nodo con el servidor objetivo. Si no existe el servidor, la función retornará NULL. Crear las estructuras necesarias.

(50 puntos)