UTN - FRBA
Departamento De Electrónica

FINAL INFORMATICA I	***Duración***	***Fecha***	***Hojas***
	90 minutos	**13/12/2023**	

Nombre y Apellido	***Nº Legajo***	***Calificación***		***Docente Evaluador***	
		número	***letras***	***Nombre***	***Firma***

Numere las hojas entregadas, y complete el casillero ***Hojas*** *con la cantidad sin incluir las del tema.*
Lea detenidamente el enunciado, su correcta interpretación forma parte de esta evaluación.

Parte Teórica

1) Enumere y explique cada una de las capas de un diagrama de Sistema Operativo del tipo unix de círculos concéntricos. ¿Cuál/les de ellas no son obligatorias?

2) Indique cuál de las siguientes afirmaciones es la correcta, y explique por qué:
   sizeof(int *) > sizeof(long int *)
   sizeof(int *) < sizeof(long int *)
   sizeof(int *) = sizeof(long int *)

3) Represente en complemento a 2 los siguientes números:
   123 -86

FINAL INFORMATICA I	Duración	Fecha	Hojas
	90 minutos	13/12/2023	

Nombre y Apellido	Nº Legajo	Calificación		Docente Evaluador	
		número	letras	Nombre	Firma

Numere las hojas entregadas, y complete el casillero ***Hojas*** *con la cantidad sin incluir las del tema.*
Lea detenidamente el enunciado, su correcta interpretación forma parte de esta evaluación.

Parte Práctica

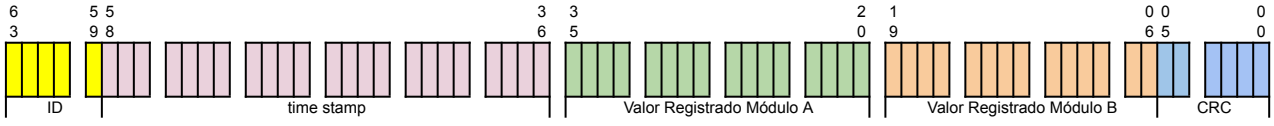
1) Se pide realizar una función que realice la conversión entre un vector de números *unsigned long de 64 bits* y los campos de una estructura llamada *cliente_t*.
La estructura estará compuesta por 4 campos:
ID: Entero sin signo de 8 bits.
timestamp: Entero sin signo de *32 bits*.
módulo A: Entero sin signo de *16 bits (unsigned short int)*.
módulo B: Entero sin signo de *16 bits (unsigned short int)*.

La lógica de conversión para completar la estructura es la siguiente:
Por cada número *unsigned long* tomará los 5 bits más significativos (del 59 al 63) para el *ID*, los siguientes 23 bits (del 36 al 58) para el *timestamp*, los siguientes 16 bits (del 20 al 35), para el valor del *módulo A* y los siguientes 14 bits (del 6 al 19) para el valor del *módulo B*.
Los últimos 6 bits corresponden al *CRC* y no serán tenidos en cuenta.

De esta manera, el prototipo será:
int ***convertir*** *(cliente_t ** **listado**, unsigned long * **datos**, int **cant**);*

*cliente_t ** **listado**:* Dirección donde guardar la dirección de inicio del vector de estructuras generado por la función.
*unsigned long * **datos**:* Dirección de inicio del vector de datos origen.
int ***cant***: Cantidad de datos a procesar.
retorna: resultado de la operación. *OK, ERRMEM.*
Si hubo error en la reserva de memoria para el vector generado, la dirección del listado debe ser NULL y no debe dejar memoria reservada.

Detalle del dato recibido en el vector datos:



FINAL INFORMATICA I	Duración	Fecha	Hojas
	90 minutos	13/12/2023	

2) Se pide realizar una función que abra el archivo binario cuyo nombre se recibe por argumento y devuelva por referencia la dirección de un vector de *unsigned long de 64 bits* y su longitud. Por retorno, deberá devolver el resultado de la operación con las constantes simbólicas definidas de *OK*, *ERRMEM* o *ERRARCH* que deberán estar disponibles para ser utilizadas en la función main.

   De esta manera, el prototipo será:

   int ***cargar*** *(unsigned long ** **datos**, int * **cant**, char * **nombre**);*

   *unsigned long ** **datos**:* Dirección donde guardar la dirección de inicio del vector de datos.
   *int * **cant***: Dirección de memoria donde guardar la cantidad de datos cargados en el vector.
   *char * **nombre***: Nombre del archivo a leer.
   retorna: resultado de la operación. *OK, ERRMEM o ERRARCH.*
   Si hubo error en el vector generado, la dirección del listado debe ser NULL y no debe dejar memoria reservada.

3) Se pide realizar una función que escriba los datos de un vector de estructuras *cliente_t* en un archivo de texto. Para ello, recibirá la dirección de memoria donde comienza el vector, su longitud y el nombre del archivo a escribir. Por retorno, deberá devolver el resultado de la operación con las constantes simbólicas definidas de *OK* o *ERRARCH* que deberán estar disponibles para ser utilizadas en la función main.

   De esta manera, el prototipo será:

   int ***guardar*** *(cliente_t * **listado**, int **cant**, char * **nombre**);*

   *cliente_t * **listado***: Dirección donde comienza el vector de estructuras.
   int ***cant***: Cantidad de elementos presentes en el vector.
   *char * **nombre***: Nombre del archivo a escribir.
   retorna: resultado de la operación. *OK o ERRARCH.*

   Un ejemplo de formato del archivo de texto podría ser:

   ```
   ID: 10      s: 4959830  Mod a: 40200      Mod b: 5944
   ```

4) Se pide realizar un programa de test de las funciones desarrolladas anteriormente.
   Para realizar la prueba, recibirá por argumento de main, el nombre del archivo binario de origen y el nombre del archivo de texto a generar.
   Luego llamará a las funciones **cargar**, **convertir** y **guardar**.

   Deberá desarrollar el archivo fuente main.c con la función main, el archivo fuente funciones.c con las tres funciones pedidas y todas aquellas que considere oportuno agregar, y el archivo fuente funciones.h que entre otras cosas deberá contener la declaración de la estructura de datos *cliente_t* y las constantes simbólicas necesarias.