**Apellidos.**.....

**Nombre** ..... DNI .....

LOGIN: exmtp001 PASSWORD:osotwuca

**NOTAS IMPORTANTES.-Lee con mucha atención antes de comenzar a trabajar.**

- **Debes tener durante todo el examen tu dni o pasaporte sobre la mesa.**
- **Debes entrar en la cuenta que figura en tu examen (*login* y *password*). No debes salir de esta cuenta durante el examen y sólo tienes permitido utilizar un editor, el compilador, el depurador y el man.**
- **Debes crear un fichero en tu cuenta de nombre *loginuco.txt* (ej. *i92romeo.txt*) con tus datos**
- **Debes realizar cada ejercicio en su directorio correspondiente. En caso contrario no se evaluará.**
- **No se podrá salir del aula durante el examen, en caso contrario el examen se entenderá como entregado.**
- **La puntuación indicada en los ejercicios es orientativa. El profesor podrá puntuar el ejercicio en conjunto para tener en cuenta la comprensión global de la asignatura.**
- **Los ejercicios deberán compilar. En caso contrario, no se evaluarán.**
- **La salida por pantalla será clara y ordenada, en caso contrario restarás un punto a tu nota final.**
- **Todos los ejercicios deben implementarse usando funciones. Excepto en las funciones específicas de entrada/salida y el *main*, no deben pedirse o mostrarse valores de variables dentro de las funciones. En caso de no cumplir estos requisitos, la nota final se verá penalizada.**
- **No se podrá hacer uso de variables globales.**
- **Los exámenes se pasarán por un software detector de copias. Se considerarán copiados todos los alumnos implicados, con la consiguiente penalización.**

# *EJERCICIOS (Hay que usar funciones en todos) 2 horas*

### (2.5 puntos) Ejercicio 1 (ELIMINATORIO)

En el directorio *ejercicio1* de tu cuenta de examen, tienes un fichero binario (*stock.bin*) con información sobre los libros que hay en una librería. La información está almacena mediante la siguiente estructura:

```
struct libro
   char titulo[256];
   char autor[256];
   float precio;
   int unidades;
};
```

Diseña y codifica una función que convierta el fichero binario a un fichero de texto. La información de cada libro se almacenará en el fichero de texto utilizando 3 líneas (título, autor y precio y unidades).

*Los pilares de la tierra*
*Ken Follet*
*24.95 15*

Ten en cuenta lo siguiente:

- El programa recibirá como argumentos en la línea de órdenes los nombre de los ficheros binario y texto.
- El programa deberá comprobar la existencia del fichero binario.
- En el directorio *ejercicio1* tienes un *visualiza.exe* que te permite visualizar el fichero *stock.bin*.

### (4 puntos) Ejercicio 2

Implementa un programa que, **utilizando funciones** realice las siguientes operaciones secuencialmente:

1. Rellenar un vector dinámico de *N* (número pedido por pantalla) valores reales con valores en el intervalo [1-10].
2. Mostrar el vector por pantalla.
3. Ordenar el vector por cualquiera de los métodos de ordenación vistos en clase.
4. Mostrar por pantalla el vector ordenado.
5. Liberar la memoria usada.

Ten en cuenta lo siguiente:

- La función para mostrar el vector será una función **recursiva.**
- El programa preguntará al usuario el sentido de la ordenación.
- La función de ordenación realizará la ordenación en uno u otro sentido mediante el uso de un **puntero a función** que se pasará como **parámetro**.
- Divide tu programa en 5 ficheros: *main.c vectores.c orden.c vectores.h orden.h.*
- Utiliza inclusión condicional de código en tus ficheros *.h*

**( 3.5 puntos) Ejercicio 3**

En el directorio *ejercicio3* de tu cuenta de examen, tienes tres ficheros que permiten trabajar con listas cuyos nodos tienen la siguiente estructura:

```
struct fraccion{
   int numerador;
   int denominador;
};
```

```
struct nodo {
   struct fraccion f;
   struct nodo * sig;
};
```

Lo ficheros son:

*visualiza.o* : código objeto de la función para visualizar una lista de fracciones.
*inserta.o*: código objeto de la función que permite insertar una nueva fracción al inicio de la lista.
*funciones.h*: tipo de dato y prototipo de las funciones anteriores.

Crea un programa que, utilizando funciones, realice las siguientes operaciones secuencialmente:

- Crear una lista de *N* fracciones (número pedido por pantalla).
- Mostrar la lista por pantalla.
- Determinar las fracciones que representan el mayor y el menor real de la lista. Para ello implementa una **única** función que, utilizando paso de parámetros por referencia, busque ambas fracciones.
- Mostrar ambas fracciones por pantalla.
- Borrar la última fracción de la lista.
- Volver a mostrar la lista.

Ten en cuenta lo siguiente:

- Utiliza las funciones proporcionadas para crear y visualizar la lista.
- Divide tu programa en 3 ficheros: *main.c listas.c listas.h.*
- Crea un **fichero makefile** con las siguientes características:
  - El ejecutable se creará a partir de *main.o*, *listas.o* y la librería *libListas.a.*
  - La librería *libListas.a* se creará a partir de *visualiza.o* e *inserta.o*.

**Hora de entrega:** **Firma:**