Junio 2003

1. En un observatorio meteorológico se tienen guardados en un fichero de tipo texto denominado “Datos.txt” los valores de lluvia mensuales de un periodo de tiempo. En el primer registro del fichero están almacenados dos números enteros que se corresponden con el número de años de la serie y con el año inicial y a continuación están guardados los datos mensuales en formato float, de forma tal que cada año ocupa una línea en el fichero, tal como se refleja debajo:

```
25      1975
23.5    23.6    45.9    34.9    25.9    34.7    34.8    12.8    89.5    12.8    12.6     5.3
34.6    45.6    56.1    32.1    1.2     23.6    7.8     7.9     15.4    16.5    12.5    17.3
.......
.......
```

Implementar un programa en C que realice de manera secuencial (sin usar menú) las siguientes acciones:

   a. Lectura de los datos de lluvia del fichero y almacenamiento de los mismos en una matriz dinámica.
   b. Almacenamiento en un vector de los datos medios mensuales de lluvia para cada uno de los meses del año.
   c. Almacenamiento en un vector de los datos anuales de lluvia para cada uno de los años de la serie.
   d. Obtención del més y del año en el cual se haya producido la precipitación máxima y la mínima.
   e. Almacenamiento de los datos de lluvia mensuales correspondientes a los años cuya precipitación anual se ainferior a 300 mm. en un fichero binario denominado “sequia.bin”.

Cada uno de los apartados se implementará en una función, y todas han de ser de tipo void.

2. En una estructura tipo lista se quiere almacenar las estaturas en cm. de una muestra de personas de una población. En cada elemento de la lista se almacenará una estatura y el número de veces que ésta se repite en la muestra de personas (frecuencia). Implementar un programa en C que posea un menú con las siguientes opciones:

   a. Dada una estatura, insertarla en orden creciente en la lista.
   b. Dada una estatura, borrar un elemento de la lista con esa estatura teniendo cuenta la frecuencia de ésta.
   c. Listar en orden decreciente las estaturas almacenadas en la lista con sus correspondientes frecuencias.
   d. Cálculo de la estatura media de los elementos almacenados en la lista.
   e. Borrado completo de la lista.

Cada uno de los apartados se implementará en un función.

Septiembre 2003

PROBLEMAS

1.- Una lista simplemente enlazada está formada por un conjunto de elementos. Cada elemento contiene un entero y una cadena de 30 caracteres. Se supone que la lista está creada y que en ella están insertados un número desconocido de elementos, siendo ***inicio*** el identificador del puntero que apunta al primer elemento de la lista.

a) Codificar una función que indique cuantos elementos hay en la lista.
b) Codificar una función que elimine de la lista aquellos elementos cuyo valor entero sea par y mayor de 50.
c) Codificar el procedimiento mas simple posible que visualice los elementos de la lista como si la misma fuera una "pila".

2.- Se desean almacenar en una matriz dinámica los datos correspondientes a las calificaciones obtenidas en una serie de asignaturas por un conjunto de alumnos.
Se supone que todos los alumnos han cursado una titulación de 6 cursos y que en cada curso hay 5 asignaturas, todas con idéntico número de créditos. En cada fila deben aparecer las 5 calificaciones de un alumno, en cada columna la calificación correspondiente a una asignatura y una tercera dimensión las calificaciones correspondientes a cada curso. En un fichero denominado alumnos.bin se encuentran almacenados los nombres de los alumnos junto con un identificador (número entero), según la plantilla struct alumno { char nombre [100]; int n;}.

a) Codificar una función que calcule el numero de estudiantes que hay en el fichero, cree la matriz necesaria e inserte las calificaciones de cada estudiante en la fila de la matriz correspondiente al identificador del alumno en el fichero.

b) Codificar una función que reciba el nombre de un alumno y devuelva en un vector las calificaciones medias del estudiante en cada uno de los 6 cursos.

c) Obtener el nombre del estudiante que posee mejor expediente académico (media de las medias de cada uno de los 6 cursos).

CUESTIONES:

1.- Se tiene definida una variable *int x*, global , y una variable *int x*, local en el interior del main. ¿Cuál tiene prioridad?

a) La global
b) La local
c) Dentro del main la local y fuera del main la global

2.- Indica cual de las siguientes afirmaciones sobre los métodos de ordenación es correcta.

a) El método quicksort es el más rápido de todos, independientemente del número de elementos a ordenar.
b) El método de selección tiene un mejor comportamiento que el quicksort cuando el número de elementos a ordenar es relativamente pequeño.
c) El método burbuja es el mejor de los métodos de orden n^2.

3.- Se tienen las siguientes declaraciones:

```
int pepe(void);
void juan(int);
```

¿Es correcta la expresión *juan(pepe())* escrita dentro de *main()*?

a) No, si se quiere pasar el valor que devuelve pepe a juan es necesario escribir *x=pepe(); juan(x);*
b) Si
c) Si, pero la expresión no debe estar en el *main()* sino en la zona de declaraciones globales.

4.- Se tienen las siguientes líneas de código. El propósito de las mismas es recoger en la dirección del campo x de la estructura, la dirección del campo y de la estructura.

```
struct dato {int x; int y;};
struct dato x, *y;
y = &x;
&(x.x)=y->y;
```

¿Cuál es correcta entre las respuestas?

a) Todo es correcto.
b) Hay dos errores. No puede ser que haya una variable “x” de estructura cuando hay un campo llamado “x” en la estructura, y el valor del campo “y” es lo que se coloca en la dirección del campo “x”.
c) Hay un error, ya que no se puede modificar la dirección del campo x.

5.- ¿Cual de las siguientes afirmaciones es correcta con relacion al siguiente esquema?

```
for (i=0;i;i=i+3)
{
        i=i-3;
}
```

a) El esquema nunca finaliza, es infinito.
b) No se entra en el esquema.
c) Se entra la primera vez y se abandona tras la primera iteración.

6.- La instrucción "*break*" dentro de un *switch*

a) Sirve para abandonar el esquema *switch* en cuanto se encuentra.
b) Es obligatorio su uso dentro de cada "*case*".
c) Si no se utiliza el esquema *switch* funciona como un esquema iterativo.

7.- El anidamiento de esquemas condicionales del tipo *if-else* dentro de un programa

a) Tiene como límite máximo 50 anidaciones dentro del mismo esquema.
b) No tiene ningún límite en cuanto al número de anidaciones.
c) Existe un límite en el número de anidaciones que depende del compilador que se utilice, aunque este límite es tan alto que en la practica no hay que preocuparse.

8.- Si *x* es una variable de tipo *int* y *p* es una variable de tipo *int* *, la instrucción *&(&x)=p*;

a) No es correcta porque debe ser *p=&(&x);*
b) No es correcta.
c) Es correcta.

9.- Una función debe recibir un parámetro que es un número real, para utilizarlo sin modificarlo, y un vector de enteros, para modificarlo. ¿Cual es el prototipo correcto?

*a) void funcion(float, int *);*
*b) void funcion(double, int *);*
c) a) y b) son correctas.

10.- El algoritmo

```
Algoritmo p(n)
Inicio
 Si n<1
        X<- 1
        Y<- 2
        Devolver (x*y)
 Sino
        X<- 1
        Y<- 2
        Devolver p(n-1)
 Finsi
Fin
```

Tiene como tiempo de ejecución

a) 2(n+1)
b) $2n^2$
c) n + 2

Junio 2004

PROBLEMAS

Copiar en vuestro directorio el fichero binario datos.bin ubicado en el siguiente subdirectorio: /home/ma1capoa/PUBLICO. Los registros del fichero tienen la siguiente estructura:

- campo ***nombre*** de 15 caracteres.
- Campo ***pais*** de 15 caracteres.
- Campo ***anyo*** de tipo entero.

Implementad un programa en C, que contenga 3 archivos (***principal.c, fichero.c y lista.c***). Para cada archivo se ha de crear su correspondiente archivo de cabecera (.h).
El programa ha de invocar de manera secuencial (no hay que hacer menú) a las siguientes funciones:

1. Función que devuelva el número de registros del fichero, sin leer todos los registros del mismo. (1 punto)
2. Función que lea todos los registros del fichero y los almacene en un vector dinámico cuyo tamaño se obtiene del apartado uno. (2 puntos).
3. Función que visualice los elementos del vector. (1 punto).
4. Función que reciba el vector de estructuras del apartado 2, y almacene los elementos del mismo en dos listas (***lista1*** y ***lista2***) ordenadas por el campo nombre, de forma tal que en la ***lista1*** estén los registro cuyo campo ***anyo*** sea impar, y en la ***lista2*** los pares. (3 puntos)
5. Función recursiva que recorra de forma inversa la ***lista1***(2 puntos).
6. Función para visualizar los elementos de la ***lista2*** (1 punto).

La función main se implementará en el archivo ***principal.c***, las funciones de los apartados 1, 2 y 3 se implementarán en el archivo ***fichero.c*** y las funciones de los apartados 4, 5 y 6 se implementarán en el archivo ***lista.c.***

Nota: No se puede hacer uso de variables globales.
Para hacer el apartado 4 se puede usar más de una función.

CUESTIONES

1. Describe brevemente como se organiza la memoria en tiempo de ejecución en un programa en C.
2. Indica en qué casos el quicksort no es el método de ordenación más rápido justificando tu respuesta.
3. Deducir el orden de complejidad del tiempo de ejecución del siguiente algoritmo:

```
Algoritmo recursivo(n; ;)
Inicio
 Si (n<=1) devolver 4
 Sino
  Devolver recursivo(n-1) + recursivo(n-1)
 Finsi
Fin
```

4. Se tiene un programa en C, con los siguiente archivos: principal.c, f1.c, f2.c y f3.c, cada uno de ellos tiene su archivo de cabecera (principal.h, f1.h, f2.h y f3.h). Se sabe que hay funciones en f1.c que invocan a funciones que están en f2.c, y que hay funciones en f2.c que invocan a funciones que están en f3.c y que el principal invoca a funciones de f1, f2 y f3. Implementa un makefile para este programa, de forma tal que se obtenga un ejecutable llamado programa.x.
5. Obtener el mismo ejecutable del apartado 4 creando una librería con las funciones (sin usar makefile).

Septiembre 2004

Se desean recoger los datos de ventas de móviles de una tienda comercial. Por cada teléfono vendido se desea guardar la marca del móvil, el modelo, el precio de venta, el precio de coste y el nombre del Operador de TELEFONIA en el que se da de alta. Defínase un tipo de datos denominado **teléfono** que conste de los campos: **marca** y **modelo** de tipo carácter con 15 de longitud, **precioventa** y **preciocoste** de tipo real y **operador** de tipo carácter con una longitud de 10.

a) Realizar un menú simple que tenga las siguientes opciones: "Introducir móvil", "Visualizar móviles", "Cargar fichero", "Calcular beneficios" y "Salir". El programa sólo deberá finalizar si señala la opción Salir, por lo que tras ejecutar cualquier opción deberá presentarse de nuevo el menú. (1 punto)
b) Para la opción "Introducir móviles" codificar una función llamada lista que se encargue de permitir introducir los datos de un solo móvil. Esta función se implementará mediante la creación-inserción en una lista doblemente enlazada. Cada vez que el usuario quiera introducir datos de un móvil deberá acceder desde el menú a esta opción. Tras introducir los datos de un móvil la función deberá mostrar todos los datos de móviles introducidos hasta ese momento comenzando por el primero.(2 puntos)
c) La opción "Visualizar móviles" deberá mostrar todos los datos de la lista comenzando por el último móvil introducido y finalizando por el primero. Esta función deberá, después de mostrar los datos, preguntar al usuario si desea generar un fichero o no, para guardar esos datos. En caso de respuesta afirmativa se deberá llamar a una función denominada "Crear_fichero" que genere un fichero de tipo binario que se llame "telefonos.bin" en el que se almacenen las informaciones de la lista. (2 puntos)
d) La opción "Cargar fichero" deberá comprobar que existe un fichero denominado "telefonos.bin" y sólo podrá ser utilizada si previamente existe ese fichero. Si ese fichero no existe aún, deberá indicarlo mediante un mensaje. Si existe ese fichero, leerá los registros cargando sólo el dato de **marca** de cada uno de los teléfonos, en un vector dinámico en el que cada elemento sea de tipo cadena, y los de preciocoste y precioventa en una matriz de dos columnas y con tantas filas como teléfonos existan en el fichero. Debes mostrar el vector y la matriz por pantalla. Si deseas probar esta opción y las de los apartados anteriores no te funcionan puedes hacer uso del fichero denominado "ejemplo.bin" que te proporcionamos.(2,5 puntos)
e) La opción "Calcular beneficios" deberá preguntar al usuario por una marca de teléfono y calcular el beneficio que esa marca de móviles ha proporcionado. Para ello deberás hacer uso del vector y la matriz generados en el apartado anterior, detectando por tanto que si el vector o la matriz no existen esta opción deberá indicarlo y no hacer nada. (2,5 puntos)

Diciembre 2004

Se desea almacenar en un fichero binario información relativa a una serie de estudiantes universitarios. La información consiste en el Primer Apellido del alumno, su DNI, y la edad actual. Para ello se especifica una plantilla formada por una cadena de caracteres de longitud 15 (para el apellido), una cadena de caracteres de longitud 9 (para el DNI) y un entero (para la edad). El nombre del fichero se le preguntara al usuario desde el programa principal y deberá abrirse en ese momento.

La realización del menú del programa principal, y la modularización realizada (archivo de cabecera, archivo de funciones y archivo principal) se valorará entre 0 y 1,5 puntos.

1.- Codificar una función que no reciba parámetros y solicite al usuario los datos de un único alumno, devolviendo esos datos al programa principal. En el programa principal deberán almacenarse los datos en el fichero. Desde el programa principal podrá llamarse a esta función de introducción de datos tantas veces como el usuario desee. Al finalizar esta opción deberá cerrarse el fichero en el programa principal. (1 puntos)

2.- Realizar también una función que reciba como parámetro el nombre del fichero y visualice todos los registros del mismo. Esta función deberá llamarse desde el programa principal. (1 punto)

3.- Codificar una función que reciba como parámetro el nombre del fichero y un entero que puede ser 1 ó 2. Esta función deberá crear una lista simple devolviendo al programa principal el apuntador a la cabeza de la lista. Si el entero que recibe es 1, la lista deberá contener sólo los apellidos de los alumnos. Si el entero recibido es 2, la lista deberá contener sólo los DNI de los alumnos. Desde el programa principal se podrá llamar a esta función tantas veces como se quiera para poder crear, si el usuario lo desea, dos listas distintas, una con los apellidos y otra con los DNI. (3,5 puntos)

4.- Codificar una función que reciba como parámetros los dos apuntadores a las listas simples creadas. En la función se le preguntará al usuario un apellido y la función deberá buscar en la lista de apellidos, para visualizar los DNI correspondientes en la otra lista (observar que pueden existir apellidos repetidos y que cada uno tiene un DNI distinto y también que el orden de ambas listas, tal y como se crean las listas, mantiene un orden entre apellidos y DNI). (3,5 puntos)

Junio y Septiembre 2005

PROBLEMAS

La famosa productora de cine MTP PICTURES S.A. está preparando una nueva película: "Pánico en las aulas X", que estrenará este próximo otoño. Ante el desastre de su última película "Pánico en las aulas IX", ha decidido contratar a los mejores actores y actrices del panorama mundial. Con este fin el becario de la productora ha recabado diferente información sobre los actores y la ha almacenado en un fichero binario llamado ***actores.bin*** ubicado en /home/in1lurom/PUBLICO. Los registros de este fichero tienen la siguiente estructura:

- campo ***nombre*** de 50 caracteres
- campo ***cache*** de tipo real (sueldo del actor)
- campo ***humano*** de tipo entero (1 si el actor es humano y 0 si no lo es)
- campo ***color*** de 15 caracteres (color de la piel)

Con el fin de agilizar los trámites de elección de actores, la productora ha contratado a un programador (**tú**) para que realice un programa en C que contenga 3 archivos (***principal.c, fichero.c y lista.c***). Para los archivos ***fichero.c*** y ***lista.c*** se han de crear su correspondiente ficheros de cabecera (.h).

Para la realización del programa se deberá tener en cuenta lo siguiente:

a) El programa invocará de manera secuencial (no hay que hacer menú) a las siguientes funciones:

   a. Se quiere saber cuántos actores componen el fichero. Implementa una función que determine el número de registros que el becario ha introducido en el fichero, sin leer todos los registros del mismo. (**0.5 puntos**)

   b. Para que el proceso se agilice, se prefiere trabajar con los datos en memoria. Crea una función que lea todos los registros del fichero y los almacene en un vector dinámico cuyo tamaño se obtiene usando la función del apartado 1. (**1 punto**)

   c. Función que muestre por pantalla todos los actores que participan en el casting, visualizando para ello todos los elementos del vector. (**0.5 puntos**)

   d. La productora no quiere gastarse mucho dinero, así que ha decido contratar a los actores más baratos. Crear una función que ordene el vector de forma creciente de acuerdo al valor de cache de cada actor, utilizando para ello cualquier método de ordenación. (**2 puntos**)

   e. Para el protagonista, el director busca un actor que no se humano. Crea una función que reciba el vector de estructuras del apartado 4 y almacene en una **lista** simplemente enlazada aquellos actores que cumplan esta condición (campo **humano** igual 0). (**2 puntos**)

   f. Por último, para que toda la información se pueda visualizar correctamente, vuelca la **lista** en un fichero de texto "casting.txt". Este fichero almacenará en cada fila los datos de un actor, separándolos por un *. Si por cualquier razón la lista estuviera vacía, el fichero contendrá el mensaje "LISTA VACÍA" (**2 puntos**)

Formato del fichero de texto

Pedro Picapiedra * 15000 * 1 * naranja
Abeja Maya * 1000 * 0 * amarilla

b) El ejecutable del programa se llamará **productora**. Para ello, implementa un makefile de forma que se obtenga el ejecutable productora. El ejecutable se podrá obtener creando una librería con las funciones (**1 punto**) o sin crearla (**0.5 puntos**).

c) Los nombres de los ficheros que utilizará el programa (actores.bin y casting.txt), se le pasarán como parámetros en la línea de ordenes. (**1 punto**).

La función main se implementará en el archivo **principal.c**, las funciones de los apartados 1, 2, 3, 4 se implementarán en el archivo **fichero.c** y las funciones de los apartados 5 y 6 se implementarán en el archivo **lista.c**

Ejercicio para subir nota

- Implementa una función extra que borre toda la lista de forma recursiva (**1 punto**). Se recomienda volcar la lista borrada a un fichero de texto (vacia.txt) para probar el funcionamiento correcto de esta función.

NOTA:

- Si no consigues hacer los apartados 1 y 2, podrás realizar el resto del examen pidiendo el número de actores y sus datos por teclado e insertándolos en el vector. De hacerlo así restarás un punto a tu nota final (Los apartados 1 y 2 se considerarán mal hechos).
- No se podrá hacer uso de variables globales.
- Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final. Si además, no estructuras el examen en los archivos indicados, restarás otro punto más a tu nota final.

CUESTIONES

1. Describe brevemente el modelo de organización de memoria que utiliza el lenguaje C, indicando la utilidad de cada una de las zonas de memoria.
2. Calcula la complejidad del siguiente algoritmo

```
Algoritmo recursivo(n;;)
Inicio
        si n<=1 entonces
                devolver 2
        sino
                devolver recursivo(n-1) + recursivo(n-1)
        finsi
fin
```

3. Compara los distintos métodos de ordenación estudiados, ordenándolos de menor a mayor eficiencia e indicando las ventajas e inconvenientes de cada uno de ellos.
4. Comenta brevemente los objetivos que se pretenden conseguir con las pruebas de la caja blanca

Diciembre 2005

PROBLEMAS

TIEMPO: 2 HORAS.

Se tiene un fichero de texto denominado “alumnos.txt” ubicado en “ “, cuyos registros poseen la siguiente estructura:
-Nombre: cadena de 15 caracteres.
-Apellidos: Cadena de 25 caracteres.
-NIF: cadena de 10 caracteres.
-Nota Media: de tipo double.

Implementad un programa en C que posea un ***menú*** con las siguientes opciones:

1. Búsqueda en el fichero de los datos de un alumno, conocido su NIF. (1.5 puntos)
2. Contabilizar el número de registros del fichero. (1.5 punto)
3. Lectura de todos los registros del fichero y almacenamiento de los mismos en una lista simple denominada ***listaAlumnos***, y visualización de los elementos de la lista. (Para este apartado se han de usar dos funciones, una para almacenar los datos en la lista, y otra para mostrar la lista) (2.5 puntos)
4. Usando la lista generada en el apartado anterior, y dado un alumno por su NIF, obtener otra lista (***listaOrdenada***) en la cual se inserten los elementos por orden decreciente de la nota y además no aparezca el alumno dado por su NIF. (3 puntos).

El menú se implementará en un archivo denominado ***principal.c***, las funciones correspondientes a los apartados 1 y 2 se implementarán en un archivo denominado ***fichero.c***, y las funciones correspondientes a los apartados 3 y 4 se implementarán en un archivo denominado ***lista.c***. También se ha de usar un archivo denominado ***cabecera.h***, para los prototipos de las funciones.
Si utilizas el makefile para obtener el ejecutable, sumarás un punto. Si controlas que el apartado 4, no se puede ejecutar si antes no se ha ejecutado el tres sumarás 1 punto.

NOTA:

- ***No se podrá hacer uso de variables globales.***
- ***Si no se usa un menú se restará un punto de la nota final.***
- ***Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final. Si además, no estructuras el examen en los archivos indicados, restarás otro punto más a tu nota final.***

TEORÍA

1. Describe en qué consiste la documentación de un programa, indicando sus tipos y los apartados de cada una de ellas.
2. Calcula la complejidad del siguiente algoritmo:

```
Algoritmo recursivo(n;;)
Inicio
si n <= 1 entonces
  devolver 2
sino
  devolver recursivo(n-1) + recursivo(n-1)
finsi
fin
```

3. Describe brevemente cómo funcionan el método burbuja y el método de ordenación por selección. Comparadlos indicando sus ventajas e inconvenientes así como el orden de su tiempo de ejecución.
4. Describid qué es un programa correcto, robusto y amigable.

Tiempo: 45 minutos.

Junio 2006

PROBLEMAS

La empresa ganadera "Paco Nejoel Del Campo S.A." que se dedica a la cría y venta de conejos, dispone de un fichero binario, denominado ***conejos2006.bin*** y que está ubicado en la cuenta, con los datos de los conejos criados en el año 2006. Los registros del fichero poseen la siguiente estructura:

- Código del conejo (cadena de 6 caracteres).
- Edad del conejo (en días) de tipo entero.
- Peso del conejo (en g.) de tipo double.

La empresa ha detectado, que debido al peso, no todos los conejos pueden ser vendidos, ya que los que no sobrepasan un determinado peso, no son deseados por los consumidores; y los que sobrepasan un determinado peso tienen un exceso de grasa y no son muy aptos para el consumo.
Implementad un programa en C, que realice secuencialmente las siguientes operaciones:

1. Paso de los registros del fichero a una lista simplemente enlazada.(**2 puntos**).
2. Cálculo del peso medio ***m*** de los conejos utilizando la lista.(**1 punto**).
3. Pasar de la lista doble a un vector dinámico aquellos conejos cuyo peso se superior a ***m -300***, e inferior a ***m + 300***, contando previamente los elementos que cumplen esta condición. Siendo ***m*** el peso medio.(**2 puntos**).
4. Ordenación de los elementos del vector en orden decreciente del peso, por el método que quieras.(**1 punto**)
5. Almacenamiento de los elementos del vector en un fichero de texto denominado ***conejosSeleccionados2006.txt***.(**1 punto**).

Para la obtención del ejecutable se creará un Makefile (1.5 puntos), y el código fuente se implementará usando los siguientes archivos (1.5 puntos): un archivo ***principal.c,*** que contendrá a la función ***main***, y los archivos ***ficheros.c, listas.c, y vectores.c,*** que contendrán las funciones correspondientes a ficheros, listas y vectores respectivamente. En caso de que una función afecte a más de un tipo (por ejemplo ficheros y listas, o listas y vectores), se puede incluir en cualquiera de los dos . Cada uno de estos tres archivos llevará su correspondiente archivo de cabecera (***.h***). Los tipos de datos para la lista simple y el vector, se implementarán en un archivo de cabecera denominado ***tipos.h.***

Nota:

- **El ejecutable se denominará examen.exe.**
- **La creación del makefile (1.5 puntos) y la división en ficheros del programa (1.5 puntos) sólo se valorará si se han hecho y funcionan correctamente los tres primeros apartados.**
- **No se podrá hacer uso de variables globales.**
- **Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final. Si además, no estructuras el examen en los archivos indicados, restarás dos puntos más a tu nota final.**

TEORIA

1. Describe cómo funciona la pila de la memoria cuando se invoca a una función recursiva.
2. Calcula la eficiencia del siguiente algoritmo:

```
Algoritmo Hanoi(n; i, j;)
Inicio
        Si n > 1 entonces
                Hanoi(n-1; i, 6-i-j)
                escribir i "->"j
                Hanoi(n-1;6-i-j,j;)
        Finsi
Fin
```

3. Comenta por qué no siempre es aplicable el método de contabilización de las frecuencias.
4. Define los siguientes conceptos:
   a. Programa correcto.
   b. Programa robusto.
   c. Programa amigable

Septiembre 2006

PROBLEMAS

Se dispone de un conjunto de mensajes almacenados en un fichero binario. Los registros del fichero poseen la siguiente estructura:

Prioridad: de tipo entero.
To: cadena de 50 caracteres
From: cadena de 50 caracteres
Subject: cadena de 100 caracteres
Msj: cadena de 255 caracteres.

Estos mensajes ha sido analizados y procesados previamente para detectar la presencia de mensajes no deseados y de virus. Implementad un programa en C, que realice secuencialmente (**NO HAY QUE HACER MENÚ**) las siguientes operaciones:

6. (**1 punto**). Calcular el tamaño del fichero (en bytes) y el número de registros del fichero. Si no se muestra el resultado por pantalla no se valorará el apartado.
7. (**2 puntos**). Paso de los registros del fichero a una lista simplemente enlazada.
8. (**0.5 puntos**). Imprimir en pantalla la lista. Este apartado sólo se valorará si funciona correctamente el apartado 2.
9. (**1.5 puntos**). Cálculo de la longitud media de los mensajes de la lista (contenido del campo **Msj**). Si no se muestra el resultado por pantalla no se valorará el apartado.
10. (**1 punto**). Contar por separado aquellos mensajes cuyo asunto (campo Subject) coincida con las cadenas [SPAM] o [VIRUS]. Si no se muestra el resultado por pantalla no se valorará el apartado.
11. (**1 punto**). Pasar aquellos elementos de la lista, cuyo asunto no tenga las cadenas [SPAM] o [VIRUS] a un fichero de texto, denominado **mensajes.txt**, en que cada mensaje tenga la siguiente estructura:
    Prioridad:
    To:
    From:
    Subject:
    Msj:

Para la obtención del ejecutable se creará un Makefile (**1.5 puntos**), y el código fuente se implementará usando los siguientes archivos (**1.5 puntos**): un archivo ***principal.c,*** que contendrá a la función ***main***, y los archivos ***ficheros.c*** *y* ***listas.c,*** que contendrán las funciones correspondientes a ficheros y listas respectivamente. En caso de que una función afecte a más de un tipo (por ejemplo ficheros y listas), se puede incluir en cualquiera de los dos. Cada uno de estos archivos llevará su correspondiente archivo de cabecera (***.h***). Los tipos de datos para la lista simple y para el fichero, se implementarán en un archivo de cabecera denominado ***tipos.h.***

Notas:

- **El ejecutable se denominará examen.exe.**
- **La creación del makefile y la división en ficheros del programa sólo se valorará si se han hecho y funcionan correctamente los tres primeros apartados.**
- **No se podrá hacer uso de variables globales.**
- **Si los nombres de los archivos no se corresponden con los indicados en el enunciado del examen restarás un punto de tu nota final.**
- **Si no estructuras el examen en los archivos indicados restarás dos puntos más a tu nota final.**
- **Aquellos apartados que se implementen sin usar funciones no puntuarán, aunque funcionen correctamente.**
- **Sólo se valorarán aquellos apartados que funcionen correctamente y que cumplan la nota anterior, en caso contrario no se mirará el código.**

TEORIA

1. Describe cómo funciona la pila de la memoria cuando se invoca a una función recursiva.
2. Calcula la eficiencia del siguiente algoritmo:

```
Algoritmo Hanoi(n; i, j;)
Inicio
        Si n > 1 entonces
                Hanoi(n-1; i, 6-i-j)
                escribir i "->"j
                Hanoi(n-1;6-i-j,j;)
        Finsi
Fin
```

3. Escribe una función en C, que reciba un vector de enteros y el número de elementos, y ordene el vector en orden decreciente, usando el método que quieras.
4. Define los siguientes conceptos:
    a. Programa correcto.
    b. Programa robusto.
    c. Programa amigable

Febrero 2004

Ejercicio.- Debes entrar en la cuenta que figura en tu examen (login y password). No puedes salir de esta cuenta durante el examen y solo puedes utilizar un editor, el compilador y el depurador. Lee con mucha atención antes de comenzar a trabajar.

TIEMPO: TRES HORAS.
Recuerda que debes estructurar desde el principio el programa para tener un archivo principal (**llámalo examen.c**) en el que se encuentre sólo el main(), otro auxiliar**, llámalo auxiliar.c**, para las funciones, y los correspondientes archivos de cabecera (**examen.h y auxiliar.h**). El nombre del ejecutable deberá ser **examen.exe. Si no lo haces así restarás un punto de tu nota final. Recuerda que no debes utilizar variables globales en ningún caso. No tienes por qué hacer un menú en el programa principal. Puedes simplemente llamar a las funciones secuencialmente. Si al final tienes tiempo y haces un menú podrás sumar 1 punto más. El menú sólo te puntuará si tienes al menos dos apartados que funcionan correctamente.**

Al final del examen tienes un ejemplo para que vayas probando tu programa apartado a apartado.

Puedes solicitar al profesor que te corrija el examen cuando lo desees, pero no podrás abandonar el aula hasta que finalice el tiempo.

Define el tipo de dato denominado ***registro*** que conste de una cadena (llamada ***nombre***) de longitud 15, un entero (llamado ***edad***) y un real (float) (llamado ***peso***).

a) En el programa principal define un vector de 5 elementos de tipo registro y llama a una función, **de nombre Introducir**, de tipo ***void*** para introducir los datos de los 5 elementos. Llama también a una función, **de nombre Imprimir**, de tipo ***void*** para imprimir los datos del vector. (1 punto)
b) Codifica una función, **llámala Calcular**, que reciba el vector, un entero por valor y un real por valor. Esta función deberá determinar qué elementos del vector tienen peso menor que el número real recibido y edad mayor que el entero recibido. Cada vez que se encuentre un elemento de este tipo deberá imprimirlo por pantalla. (2 puntos)
c) Codifica una función, **llámala Datosmatriz**, que reciba el vector, una matriz ***m1 de tipo float*** de dimensión 2 x 5 y otra ***m2 de tipo float*** de dimensión 5 x 2 y no devuelva nada. **Utilizando esquemas iterativos** en la primera matriz deberás colocar en la primera fila las edades y en la segunda los pesos de todos los elementos del vector. En la segunda matriz deberás colocar en la primera columna los pesos y en la segunda las edades de todos los elementos del vector. En el programa principal deberás mostrar todos los elementos de cada una de estas 2 matrices. (2 puntos)
d) Codifica una función, **llámala Buscacadenas**, que reciba un entero y el vector y que encuentre todas las cadenas (son campos de cada elemento del vector) de los elementos del vector que tengan longitud igual al entero recibido seleccionando entre ellas la menor (orden alfabético). La función deberá devolver el elemento del vector correspondiente a la cadena encontrada, con los requisitos anteriores. El programa principal deberá imprimir este resultado.(3 puntos)

e) Codifica una función, **llámala Producto**, que reciba las dimensiones de cada una de las matrices (4 valores enteros) y las dos matrices, pasando un puntero simple que contenga la dirección del primer elemento de cada matriz, y calcule el producto, si es posible, mostrando por pantalla el resultado. Llama a esta función desde el programa principal de forma que reciba las dos matrices del apartado c). Recuerda que para multiplicar dos matrices, el elemento de la matriz producto que ocupa la fila i y la columna j se obtiene multiplicando los valores de la fila i de la primera matriz por los valores de la columna j de la segunda y sumando esos productos(2 puntos).

Caso de prueba.-

Introduce como elementos del vector los siguientes:

Nombre: Rafael Edad: 18 Peso: 65.4
Nombre: Manuel Edad: 22 Peso: 75.5
Nombre: Juan Luis Edad: 40 Peso: 80.8
Nombre: Casimiro Edad: 34 Peso: 74.2
Nombre: Jose Edad: 56 Peso: 68.5

En el apartado b) puedes probar pasando como parámetros:

1) entero: 25, real: 70.5.

Debe salir: Nombre: Jose Edad: 56 Peso: 68.5

2) entero: 35, real: 85.2

Deben salir: Nombre: Juan Luis Edad: 40 Peso: 80.8
Nombre: Jose Edad: 56 Peso: 68.5

3) entero: 20, real: 60.5

No debe salir ninguno.

En el apartado c) las matrices que salen son:

Primera: 18 22 40 34 56
65.4 75.5 80.8 74.2 68.5
22
40
74.2 34
56

Segunda: 65.4 18
75.5
80.8
68.5

El apartado d) puedes probarlo con:

Parámetro entero: 6.
Debe encontrar Rafael y Manuel. Como Manuel es menor que Rafael debe devolver el elemento segundo del vector y el programa principal mostrara:
Nombre: Manuel Edad: 22 Peso: 75.5

El apartado e) debe mostrar la siguiente matriz por pantalla:

12429.00 6700.00
26703.93 12429.00

Si quieres probar con otros datos este apartado, observa que sean los que sean los que introduzcas en las matrices del apartado c) el resultado tiene que dar los mismos valores para los elementos de la diagonal principal.

Febrero 2005

NOTA IMPORTANTE.- Debes entrar en la cuenta que figura en tu examen (login y password). No puedes salir de esta cuenta durante el examen y solo puedes utilizar un editor, el compilador y el depurador. Lee con mucha atención antes de comenzar a trabajar.

TIEMPO: 2 HORAS 30 MINUTOS.
Recuerda que debes estructurar desde el principio tus ejercicios para tener un archivo principal (**llámalo examen.c**) en el que se encuentre sólo el main(), otro para las funciones (llámalo **funciones.c**), el correspondientes archivo de cabecera (**funciones.h**). El nombre del ejecutable deberá ser **examen.exe. Si no lo haces así restarás un punto de tu nota final. Recuerda que no debes utilizar variables globales en ningún caso. No tienes por qué hacer un menú en el programa principal. Puedes simplemente llamar a las funciones secuencialmente. Si al final tienes tiempo y haces un menú podrás sumar 1 punto más. El menú sólo te puntuará si tu nota es al menos de cuatro puntos.**

NO PODRÁS ABANDONAR EL AULA HASTA QUE FINALICE EL TIEMPO.

EJERCICIO 1 (3.5)
Se dispone de un vector de enteros de tamaño máximo 10:

d) (1.5). Codifica una función de tipo void, llámala **LeeVector,** que lea desde teclado números enteros y los introduzca en el vector. La entrada de números finalizará cuando se lea un valor negativo, que no deberá considerarse, o cuando se hayan introducido 10 elementos. La función deberá devolver el tamaño final del vector (número de elementos en el vector).

e) (2.0). Codifica una función de tipo void, llámala **Busqueda**, que reciba el vector de números enteros, su tamaño y dos parámetros de tipo entero pasados por referencia. La función deberá determinar, de todos los valores del vector, cual es el número par más pequeño y el impar más grande y devolverlos usando los parámetros pasados por referencia. El programa principal imprimirá estos dos valores.

Ejemplo de prueba:
Elementos del vector: 7, 26, 3, 2, 15, 18, 42, 111
Debe salir: Número par: 2 Número impar: 111

EJERCICIO 2 (3.0)
Un palíndromo es una cadena de caracteres que se lee igual de derecha a izquierda que de izquierda a derecha, independientemente de los espacios en blanco que contenga. Por ejemplo:

"*yo hago yoga hoy*"
"*dabale arroz a la zorra el abad*"

Escribe una función **palindromo** en C para averiguar si una cadena de caracteres es un palíndromo. La función acepta como parámetro una cadena de caracteres y devuelve 0 si la cadena no es palíndromo y 1 si lo es. Suponer un tamaño máximo de cadena de 100 caracteres (incluído el carácter de fin de cadena).

EJERCICIO 3 (3.5)
Se tiene una matriz real (float), con un máximo de 10 filas y 10 columnas. Codifica los siguientes modulos:

a) (0.75). Una función, llámala **LeeMatriz,** que lea de teclado los elementos de la matriz. La función deberá preguntar al usuario el número de filas y columnas que tendrá la matríz y posteriormente devolverlos al programa principal para utilizarlos posteriormente.
b) (0.75). Una función, llámala **ImprimeMatriz**, que reciba las dimensiones de la matriz (dos enteros) y la matriz, e imprima por pantalla los elementos que la componen.
c) (2.0). Una función, llámala **Inversión**, que reciba la matriz y las dimensiones de la misma (dos enteros), y realice una inversión de la matriz por columnas. La función deberá intercambiar la primera columna de la matriz por la última, la segunda por la penúltima y así sucesivamente hasta llegar al centro de la matriz. La inversión se hará sobre la misma matriz.

Ejemplos de prueba:

- Matrices con un número de filas impar

1	2	3		3	2	1
4	5	6	=> *salida* =>	6	5	4
7	8	9		9	8	7
10	11	12		12	11	10

- Matrices con un número de filas par

5	6	7	8		8	7	6	5
3	4	2	1	=> *salida* =>	1	2	4	3

Febrero 2006

Una empresa posee datos relativos a la medición de la superficie de parcelas agrarias que se dedican a cultivar árboles de cualquier tipo. Para cada árbol se dedica un superficie cuadrada de lado k, excepto para los que están ubicados en los límites de la parcela (árboles exteriores) a los que se dedica la mitad de la superficie del cuadrado. Los datos que posee sobre cada parcela son:

- nombre del propietario de la parcela (máximo diez caracteres sin espacios en blanco).
- identificador de la parcela (código formado por 6 caracteres de los que los 2 primeros son letras minúsculas del abecedario y los 4 siguientes dígitos entre el 0 y el 9).
- Número de árboles interiores (de tipo entero).
- Número de árboles exteriores (de tipo entero).
- Lado del cuadrado k (de tipo real).

Construye un programa que sea capaz de gestionar los datos que se desean guardar de acuerdo a las siguientes especificaciones:

1) El numero máximo de parcelas a considerar será de 15. Utiliza por tanto un vector de 15 elementos.
2) De cada parcela se deberá guardar el valor del lado del cuadrado asignado a cada árbol, el número de árboles interiores a la parcela, el número de árboles exteriores a la parcela, el nombre del propietario y el identificador de la parcela.

   Inicializa todos los elementos del vector que va a almacenar los datos de las parcelas haciendo que el campo correspondiente al valor del lado del cuadrado sea -1.
   Realiza un menú en el programa principal, con las opciones que se indican a continuación, que deberá permanecer en funcionamiento hasta que el usuario señale una opción inexistente. **(1 punto)**

Deberán existir las siguientes opciones:

a) **Introducir datos**. Diseña una función llamada **introducir_datos** que reciba el vector y el número de parcelas que hay introducidas en ese momento. Pida al usuario todos los datos de una parcela, devolviendo el número de parcelas totales introducidas. En el programa principal deberás primero informar de cuantas parcelas hay introducidas; y si todavía se pueden almacenar más, llamar a la función especificada. Si ya no se pueden introducir más deberás informar de ese hecho al usuario. **(2 puntos)**

b) **Imprimir una parcela**. Diseña una función llamada **imprimir** que reciba la posición de una parcela del vector (número del elemento del vector entre 0 y 14) e imprima todos los datos de la parcela. Desde el programa principal deberás imprimir todas las parcelas que tengan datos ya introducidos usando la función implementada. **(2 puntos)**

d) **Buscar por propietario.** Diseña una función llamada **buscar_parcela** que reciba como parámetro el nombre de un propietario y que compruebe si existe alguna parcela perteneciente a ese propietario. En caso afirmativo la función deberá imprimir los datos correspondientes a esa parcela y además proporcionar la superficie de la parcela. En caso negativo deberás informar de que no existe esa parcela. **(2,5 puntos)**

e) **Buscar por identificador**. Diseña una función llamada **buscar** que reciba una cadena de dos caracteres alfabéticos y que imprima los datos de todas las parcelas que en su identificador posean esos dos caracteres. **(2,5 puntos)**

El programa ejecutable se denominará examen.exe. Si el programa que realices está estructurado de la forma siguiente:
1) Un archivo examen.h en donde se encuentren todas las declaraciones necesarias
2) El programa principal en un archivo llamado examen. c
3) Las funciones pedidas en un archivo llamado funciones.c.
obtendrás 1 punto mas en tu calificación.

Prototipo de examen primer parcial

Realiza el siguiente ejercicio siguiendo las indicaciones concretas de cada apartado.
Ejercicio.- Debes entrar en la cuenta que figura en tu examen (login y password). No puedes salir de esta cuenta durante el examen y solo puedes utilizar un editor, el compilador y el depurador. Lee con mucha atención antes de comenzar a trabajar.
TIEMPO: TRES HORAS.
Recuerda que debes estructurar desde el principio el programa para tener un archivo principal (**llámalo examen.c**) en el que se encuentre sólo el main(), otro auxiliar**, llámalo auxiliar.c**, para las funciones, y los correspondientes archivos de cabecera (**examen.h y auxiliar.h**). El nombre del ejecutable deberá ser **examen.exe. Si no lo haces así restarás un punto de tu nota final. Recuerda que no debes utilizar variables globales en ningún caso. No tienes por qué hacer un menú en el programa principal. Puedes simplemente llamar a las funciones secuencialmente. Si al final tienes tiempo y haces un menú podrás sumar 1 punto más. El menú sólo te puntuará si tienes al menos dos apartados que funcionan correctamente.**
Al final del examen tienes un ejemplo para que vayas probando tu programa apartado a apartado.
Puedes solicitar al profesor que te corrija el examen cuando lo desees, pero no podrás abandonar el aula hasta que finalice el tiempo.
Define el tipo de dato denominado ***registro*** que conste de una cadena (llamada ***nombre***) de longitud 15, un entero (llamado ***edad***) y un real (float) (llamado ***peso***).

f) En el programa principal define un vector de 5 elementos de tipo registro y llama a una función, **de nombre Introducir**, de tipo ***void*** para introducir los datos de los 5 elementos. Llama también a una función, **de nombre Imprimir**, de tipo ***void*** para imprimir los datos del vector. (1 punto)
g) Codifica una función, **llámala Calcular**, que reciba el vector, un entero por valor y un real por valor. Esta función deberá determinar qué elementos del vector tienen peso menor que el número real recibido y edad mayor que el entero recibido. Cada vez que se encuentre un elemento de este tipo deberá imprimirlo por pantalla. (2 puntos)
h) Codifica una función, **llámala Datosmatriz**, que reciba el vector, una matriz ***m1 de tipo float*** de dimensión 2 x 5 y otra ***m2 de tipo float*** de dimensión 5 x 2 y no devuelva nada. **Utilizando esquemas iterativos** en la primera matriz deberás colocar en la primera fila las edades y en la segunda los pesos de todos los elementos del vector. En la segunda matriz deberás colocar en la primera columna los pesos y en la segunda las edades de todos los elementos del vector. En el programa principal deberás mostrar todos los elementos de cada una de estas 2 matrices. (2 puntos)
i) Codifica una función, **llámala Buscacadenas**, que reciba un entero y el vector y que encuentre todas las cadenas (son campos de cada elemento del vector) de los elementos del vector que tengan longitud igual al entero recibido seleccionando entre ellas la menor (orden alfabético). La función deberá devolver el elemento del vector correspondiente a la cadena encontrada, con los requisitos anteriores. El programa principal deberá imprimir este resultado.(3 puntos)
j) Codifica una función, **llámala Producto**, que reciba las dimensiones de cada una de las matrices (4 valores enteros) y las dos matrices, pasando un puntero simple que contenga la dirección del primer elemento de cada matriz, y calcule el producto, si es posible, mostrando por pantalla el resultado. Llama a esta función desde el programa principal de forma que reciba las dos matrices del

apartado c). Recuerda que para multiplicar dos matrices, el elemento de la matriz producto que ocupa la fila i y la columna j se obtiene multiplicando los valores de la fila i de la primera matriz por los valores de la columna j de la segunda y sumando esos productos(2 puntos).

Introduce como elementos del vector los siguientes:

Nombre: Rafael	Edad: 18	Peso: 65.4
Nombre: Manuel	Edad: 22	Peso: 75.5
Nombre: Juan Luis	Edad: 40	Peso: 80.8
Nombre: Casimiro	Edad: 34	Peso: 74.2
Nombre: Jose	Edad: 56	Peso: 68.5

En el apartado b) puedes probar pasando como parámetros:

4) entero: 25, real: 70.5.

Debe salir: Nombre: Jose Edad: 56 Peso: 68.5

5) entero: 35, real: 85.2

Deben salir: Nombre: Juan Luis Edad: 40 Peso: 80.8

Nombre: Jose Edad: 56 Peso: 68.5

6) entero: 20, real: 60.5

No debe salir ninguno.

En el apartado c) las matrices que salen son:

Primera: 18 22 40 34 56 Segunda: 65.4 18

65.4 75.5 80.8 74.2 68.5 75.5 22

80.8 40

74.2 34 68.5 56

El apartado d) puedes probarlo con:

Parámetro entero: 6.

Debe encontrar Rafael y Manuel. Como Manuel es menor que Rafael debe devolver el elemento segundo del vector y el programa principal mostrara:

Nombre: Manuel Edad: 22 Peso: 75.5

El apartado e) debe mostrar la siguiente matriz por pantalla:

12429.00 6700.00

26703.93 12429.00

Si quieres probar con otros datos este apartado, observa que sean los que sean los que introduzcas en las matrices del apartado c) el resultado tiene que dar los mismos valores para los elementos de la diagonal principal.