DICIEMBRE 2009

NOTAS IMPORTANTES.-

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 puedes salir de esta cuenta durante el examen y sólo puedes utilizar un editor, el compilador y el depurador.**
- **Lee con mucha atención antes de comenzar a trabajar.**
- **Debes crear un fichero en tu cuenta cuyo nombre será: *loginuco.txt* (ej. *i62romeo.txt*) y contendrá tu nombre, apellidos y dni. En caso contrario no se corregirá tu examen.**
- **Debes realizar cada ejercicio en su directorio correspondiente. En caso contrario no se evalurá.**
- **La puntuación indicada en los apartados es orientativa. El profesor podrá puntuar el ejercicio en conjunto para tener en cuenta la comprensión de la asignatura.**

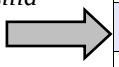
TIEMPO: 4 h

EJERCICIOS

(3 puntos) Ejercicio 1 -

Se dice que una matriz tiene un punto de silla si alguna posición de la matriz es el menor valor de su fila, y a la vez el mayor de su columna.

Punto de silla



1	3	6
7	8	9
2	4	1

Para calcular los puntos de silla de una matriz se necesitan la matriz original y una matriz auxiliar donde se marcarán los puntos de silla. El procedimiento para calcular los puntos de silla de una matriz es el siguiente:

1. Leer la matriz original
2. Inicializar la matriz auxiliar a 0
3. Calcular los mínimos de cada fila de la matriz original y sumar 1 a las posiciones que ocuparían los mínimos en la matriz auxiliar.
4. Calcular los máximos de cada columna de la matriz original y sumar 1 a las posiciones que ocuparían los máximos en la matriz auxiliar
5. Los elementos de la matriz auxiliar con valor igual a 2 son punto de silla de la matriz original.

0	0	0
0	0	0
0	0	0

(4) Máximos de cada columna

(2) Inicializar matriz auxiliar

1	0	0
1	0	0
0	0	1

(3) Mínimos de cada fila

1	0	0
2	1	1
0	0	1

Escribe un programa que calcule cuántos puntos de silla tiene una matriz. Para ello implementa las siguientes funciones sobre matrices estáticas de tipo entero (considera que el programa podrá trabajar como máximo con matrices de 20x20).

(0.25 puntos) void rellenaMatriz(int matriz[][MAX], int nFil, int nCol). Función que rellene una matriz con valores positivos leídos desde teclado. Los valores deberán estar en el intervalo [1-50].

- **(0.25 puntos) void imprimeMatriz(int matriz[][MAX], int nFil, int nCol).** Función que imprime una matriz por pantalla.
- **(0.25 puntos) void inicializaMatriz(int auxiliar[][MAX], int nFil, int nCol)**. Función que inicialice todos los elementos de una matriz a 0.

(1 punto) void minimosFila(int datos[][MAX], int auxiliar[][MAX], int nFil, int nCol). Función que calcula el mínimo de cada fila de la matriz *datos* y suma 1 a la casilla correspondiente de la matriz *auxiliar.*

- **(1 punto) int maximosColumna (int datos[][MAX], int auxiliar[][MAX],int nFil, int nCol)**. Función que calcula el máximo de cada columna de la matriz *datos* y suma 1 a la posición correspondiente en la matriz *auxiliar*.
- **(0.25 puntos) int puntosSilla(int auxiliar[][MAX], int nFil, int nCol)**. Función que devuelva el número de puntos de silla que tiene una matrz (elementos con valor igual a 2).

Deberás estructurar tu programa en dos ficheros, *funciones.c* (contendrá las funciones descritas anteriormente) y *funciones.h* (contendrá los prototipos de las funciones). El fichero *main.c* que se encuentra en tu cuenta (en el directorio ejercicio1), contiene las llamada a las funciones que has implementado; utilízalo para probar tu código.

(3 puntos) Ejercicio 2

El ayuntamiento de Niscaler ha organizado un concurso para elegir el cartel que anunciará las fiestas del pueblo este año. Para evitar favoritismos, a cada participante se le ha asignado un *id* que será el que identifique de forma anónima al cartel. De esta forma, el jurado podrá asignar a cada cartel la posición en la que ha quedado sin necesidad de conocer a su autor. Toda esta información se almacena en dos ficheros:

- Un fichero binario que almacena los datos personales de los participantes, en orden de llegada, usando la siguiente estructura:

```
!  tusvdu!dbsufm|!
!!!}dibs!ujuvmp!¥61^<!
!  !!!dibs!bvups!¥261^<!  !     !     !     !!
!!!}jou!je`dbsufm<!
!!!}jou!qptjdjpo<!
   }
```

- Un fichero de texto que almacena (en cada línea) el identificador del cartel y la posición en la que ha quedado, ordenados del mejor al peor
- Los datos de los dos ficheros se encuentran en diferente orden. El fichero binario almacena los carteles por orden de llegada y el fichero de texto almacena los identificadores en función de la posición obtenida.
- Los dos ficheros tienen el mismo número de registros.
- Incialmente, en el fichero binario, el campo *posicion* de los registros está inicializado a -1.

Se quiere guardar en un fichero de texto el título y el autor de los carteles junto con la posición en la que ha quedado el cartel. Para ello, escribe un programa que realice las siguientes operaciones de **forma secuencial,** usando funciones para ello:

a) **(0.25 puntos)** Calcular el número de carteles que hay en el fichero binario sin recorrerlo.

b) **(0.75 puntos)** Almacenar los registros del fichero binario en un vector dinámico (*informacion*) y mostrarlos por pantalla.

c) **(0.75 puntos)** Almacenar el fichero de texto en un vector dinámico (*resultado*) y mostrarlo

por pantalla
d) **(0.75 puntos)** Asignar a cada cartel almacenado en el vector *información*, la posición en la que ha quedado según el vector *resultado*.
e) **(0.25 puntos)** Almacenar el vector *informacion* en un fichero de texto

Los **nombres de los ficheros serán pasados al programa principal como argumentos en línea de órdenes (0.25 puntos)** en el momento de ejecutar el programa. El primer fichero será el fichero binario con los datos de los carteles, el segundo el fichero de texto con los resultados del concurso, y el tercero, el fichero de texto que almacenará la información de los carteles una vez incluida la posición que han obtenido.

Estructura tu programa en los siguientes ficheros, *principal.c* (contendrá el *main*), *ficheros.c* (funciones relacionadas con los ficheros), *varias.c.* (resto de funciones, reserva, ...) y los correspondientes .h (*ficheros.h* y *varias.h*)

NOTAS
Aquel que no sea capaz de leer los ficheros, podrá leer los datos desde teclado. En caso de hacerlo así, restará 0.75 a la nota de su ejercicio.

(3 puntos) Ejercicio 3
Un polinomio es una expresión algebráica de la forma:

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0$$

A cada $a_i x^i$ se le denomina *término*, siendo a_i el coeficiente del término e i el exponente del término. Algunos ejemplos de polinomios son:

$$(1) 2x + 3$$
$$(2) x^3 + 7x^2 + 3x + 9$$
$$(3) 2x^8 + x^3 + 6x$$

Un polinomio se puede representar como una lista enlazada. El primer nodo de la lista representa el primer término del polinomio, el segundo nodo el segundo término del polinomio, y así sucesivamente. Cada nodo representa un término del polinomio y tiene como campo dato el coeficiente del término (*a*) y el exponente (*e*). Escribe un programa que permita: (**0.5 puntos** para el global del ejercicio)
1. Crear un polinomio. El programa preguntará al principio cuántos términos tendrá el polinomio
2. Obtener una tabla de valores de un polinomio para valores de x = 0.0, 0.5, 1.0, 1.5, ..., 5.0

Para el polinomio (1) tendríamos la siguiente salida:
(x=0.0, 3), (x=0.5, 4), (x=1.0, 5), (x=1.5, 6), ..., (x=5.0, 13)

Implementa para ello las siguientes funciones
- **(0.5 puntos)** *anyadeTérmino*. Inserta (por delante) un nuevo término en el polinomio.
- **(0.75 puntos)** *evaluaPolinomio*. Evalúa el polinomio para un valor concreto de *x*
- **(0.5 puntos)** *muestraPolinomio*. Muestra por pantalla el polinomio
- **(07.5 puntos)** *eliminaTermino*. Elimina, si existe, el término de exponente *E* (parámetro de la función)

(1 puntos) Ejercicio 4
La empresa, AECAMTP.SA está realizando un proyecto informático. Para el desarrollo de este proyecto, los programadores están haciendo uso de los siguientes ficheros:
imagenes.c – imagenes.h
funciones relacionadas con el procesamiento de imágenes y sus prototipos
memoria.c – memoria.h
funciones relacionadas con la reserva dinámica de memoria y sus prototipos

fichero.c – fichero.h

funciones relacionadas con la E/S de datos en archivos y sus prototipos

pMatricial.c – matricial.h

main, funciones específicas para implementar el método de cifrado matricial y sus prototipos

pVernam.c – vernam.h

main, funciones específicas para implementar el método de cifrado vernam y sus prototipos

El resultado final del proyecto serán dos ejecutables (*pVernam.exe y pMatricial.exe*) que permitirán cifrar imágenes con dos métodos clásicos de cifrado de imágenes. Como programador de esta empresa y para facilitar el proceso de compilación, se te pide que crees un fichero *makefile*. Las características de este fichero son:

- **(0.25 puntos)** Construirá el ejecutable *pVernam.exe* a partir de *pVernam.c* y los ficheros de funciones (imagenes.c, memoria.c, fichero.c)
- **(0.25 puntos)** Construirá el ejecutable *pMatricial.exe* a partir de *pMatricial.c* y los ficheros de funciones (imagenes.c, memoria.c, fichero.c)
- **(0.25 puntos)** Permitirá generar los dos ejecutables en una única llamada de la orden *make.*

El fichero *makefile* que construyas debe funcionar correctamente **(0.25 puntos)**. Para probarlo puedes utilizar los ficheros que se encuentran en el directorio **ejercicio4**.

Notas:

Para que los ejercicios sean puntuados, han de funcionar correctamente y han de seguir las especificaciones reseñadas en cada uno de ellos.

- **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 contrario no se evaluará el ejercicio aunque funcione correctamente.**
- **La salida por pantalla será clara y ordenada, en caso contrario restarás un punto a tu 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.**
- **Aquellos apartados que se implementen sin usar funciones no puntuarán, aunque funcionen correctamente.**
- **No se podrá hacer uso de variables globales.**
- **No olvides firmar tu examen y poner la hora de entrega.**