Fundamentos de Informática 1º IIND
Examen Final: 14 de Junio de 2007

Apellidos:	**Nombre:**
Grupo: □ **A** □ **B** □ **C** □ **D** □ **E** □ **F**	

Advertencias:

1. Duración del examen 2 horas y 30 minutos
2. No desgrape el cuadernillo del examen.
3. Puede utilizar lápiz o bolígrafo indistintamente.
4. Las funciones sólo pueden tener un `return`.
5. Sólo puede realizar operaciones de entrada/salida (`printf()`, `scanf()`, `gets()`, etc,) en la función `main()`, salvo que el problema pida funciones específicas para leer datos o para mostrar resultados.
6. No puede utilizar `exit`, `continue`, ni `break` (salvo en la instrucción `switch()`).
7. No puede utilizar variables globales.

Calificación:

Códigos cortos (5 puntos)	**Programa 1 (1,5 puntos)**	**Programa 2 (1,5 puntos)**	**Programa 3 (2,0 puntos)**	**Total**

página en blando

Códigos cortos (5 puntos)

1. Manejo de vectores **(1 punto)**

Escribe una función cuyo prototipo es:

```
void Intercalar (int vector[], int num_elem, int num_intercalar);
```

que reciba un vector numérico para el que se han reservado 100 posiciones de forma estática, el número de elementos realmente utilizados por el vector (`num_elem`) y un nuevo valor a insertar en el vector (`num_intercalar`).

Se sabe que el vector se encuentra ordenado de menor a mayor y se trata de insertar el nuevo valor de manera que el vector siga ordenado. También se sabe que el vector tiene longitud suficiente para insertar el nuevo elemento.

2. Memoria dinámica **(1 punto)**

En una biblioteca, se quiere extraer el listado de los libros de un autor determinado. Escribir la función `ExtraerAutor` de acuerdo al siguiente prototipo:

```
int ExtraerAutor(t_libro vec[], int nun_libros, char nombre_autor[]);
```

donde `vec` es un vector de estructuras que contiene la información de todos los libros de la biblioteca, `num_libros` es el número de libros, y `nombre_autor` es el nombre del autor buscado.

La estructura `t_libro` contiene los campos siguientes: `Autor` (cadena de 25 caracteres), `Titulo` (cadena de 25 caracteres), `ISBN` (cadena de 14 caracteres), `Editor` (cadena de 20 caracteres), `Fecha_de_adquisicion` (estructura de tipo `t_fecha`), y `Precio` (tipo `double`)

La función debe extraer todos los libros de la biblioteca escritos por el autor `nombre_autor`, y crear un vector dinámico de estructuras de tipo `t_autor` donde almacenarlos.

La estructura de tipo `t_autor` contiene sólamente los campos `Autor`, `Titulo` e `ISBN`

Antes de que la función termine y devuelva el control al programa principal, deberá llamar a la función `Mostrar` (que no hace falta programar) para mostrar el vector dinámico de tipo `t_autor`.

En caso de error al asignar memoria dinámica, la función debe volver -1 y en caso contrario debe devoler 0. Se recuerda que la función no puede utilizar `printf` y que sólo se permite un `return`.

3. Expresiones lógicas **(0,5 punto)**

Indicar cuál sería la salida del siguiente código:

```
int i=2, n=87;
int j;
float x,z;
t_book a[100];  //t_book es una estructura definida anteriormente con typedef

x=n+1/2;

j=n+1/2;
if ( !(n>i) ) a[0].publication_year=0;
else a[0].publication_year=1900+n;
z=n%10;
printf("i=%d, n=%d, j=%d, year=%d\n", i,n,j,a[0].publication_year);
printf("x=%f, z=%f\n",x,z);
```

SALIDA:

---

4. Punteros **(0,5 punto)**

Un programa en C contiene las siguientes instrucciones:

```
int i, j= 15;
int *pi, *pj;
pj=&j;
*pj=j+15;
i=*pj+5;
pi=pj;
*pi=i+j;
```

…

Indicar el valor final de las variables: `i=.....  j=..... *pi=..... *pj=.....`

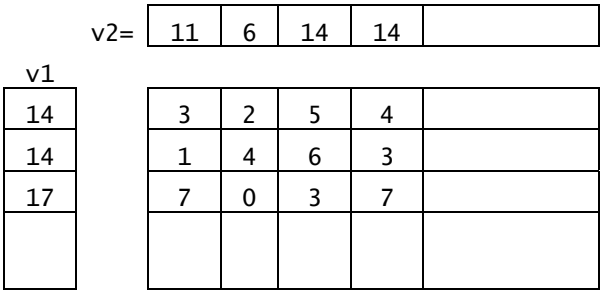
5. Cadenas de caracteres **(1 punto)**

Escribir una función que reciba como argumento una frase (cadena de caracteres) y devuelva un valor entero. El valor devuelto debe ser la longitud de la palabra más larga de la frase, sabiendo que el espacio (' ') es el único caracter separador de palabras.

6. Matrices **(0,5 punto)**

Dada una matriz numérica de tipo real de dimensiones MxN parcialmente rellena, y dos vectores numéricos reales de dimensiones suficientes (`v1` y `v2`), desarrollar la función `CalcularSumas()` que calcula las sumas de todos los elementos de cada fila y las guarda en el vector `v1`, y las sumas de cada columna y las guarda en el vector `v2`. El prototipo de la función es:

```
void CalcularSumas(float matriz[][N], int fil, int col, float v1[], float v2 []);
```



7. Corregir fallos **(0,5 punto)**

Sabiendo que el prototipo de la función `Calcular()` es correcto, indicar sobre el código cuáles son los fallos del siguiente programa

```
#include <stdio.h>
typedef struct{
   int codigo;
   float precio;
}T_producto;

void Calcular(T_producto *p);

int main()
{
   T_producto prod;

   Calcular(prod);
   printf("El codigo=%d, el precio=%.1f",prod.codigo,prod.precio);
   return(0);
}

void calcular(T_producto *p)
{
   p.codigo=100;
   p.precio=10.5;
}
```

Programa 1 (1,5 puntos)

Escribir un programa para realizar ciertas transformaciones de cadenas de caracteres. El programa realizará las siguientes operaciones:

- Declarar dos vectores de tipo `char`, la frase original (`frase`) y la frase resultante (`result`)
- Leer una frase original (`frase`) mediante la función `gets()`
- Mostrar la frase leída
- Llamar a función `Transformar`
- Mostrar el resultado (`result`)

La función `Transformar` debe tener el siguiente prototipo:

```
void Transformar(char entrada[], char salida[]);
```

en esta función se analizará la frase de entrada y se creará la frase de salida. En principio se irá copiando letra por letra, salvo que ocurra alguno de los siguientes casos especiales:

- Si la frase de entrada contiene dos letras `'s'` consecutivas, sólo se copiará una letra `'β'`
- Si la frase de entrada contiene la letra `'a'` seguida de una `'e'`, sólo se copiará una letra `'æ'`

Ejemplo: Si la frase de entrada es "`Vaenge Hoch Strasse`" la frase resultante debe ser "`Vænge Hoch Straβe`"

NOTA, en el código se puede asignar directamente el caracter `'β'` y el caracter `'æ'` sin preocuparse de cuál es el valor en la tabla ASCII, ISO-8859-1 ó ISO 8859-15. Se valora positivamente que los casos especiales no provoquen un fallo en la función `Transformar`, como por ejemplo que la cadena esté vacía.

Programa 2 (1,5 puntos)

Como parte del mantenimiento de la aplicación de Personal de una gran compañía inmobiliaria te han asignado el mantenimiento del fichero binario de vendedores denominado `comerciales.dat`. Los registros están clasificados por orden alfabético de acuerdo con los apellidos del vendedor, y contienen los siguientes campos:

Nombre (60 caracteres que contiene el nombre en formato "Apellidos, Nombre");
DNI (10 caracteres);
Edad (entero)
Sexo (1 carácter)
Fecha de incorporación (estructura de tipo `t_fecha`, con fechas tipo `short int` DD-MM-AA)

Se pide:

Escribir una función `int AltaVendedor (T_VENDEDOR vendedor)` que reciba los datos de un vendedor nuevo y, de acuerdo con sus apellidos y nombre, añada el nuevo registro en el sitio correspondiente de `comerciales.dat` de manera que se mantenga el orden alfabético. La función devolverá un código de resultado de la operación, ya que hay que contemplar la posibilidad de que se pretenda dar de alta un vendedor que ya figura en el fichero. La función devolverá 1 en caso de éxito y 0 en caso de vendedor duplicado.

En la solución codificada no está permitido cargar el fichero completo en un vector de estructuras ya que este ejercicio tiene como objetivo utilizar la función `fseek( ).`

Aunque sólo se pide codificar la función `AltaVendedor`, el alumno puede programar otras funciones auxiliares que contribuyan a mejorar la modularidad del código.

Programa 3 (2 puntos)

Todos los movimientos de cuentas que tienen lugar durante un día de trabajo en el Royal Bank of Lapland, se almacenan en un fichero binario `Movimientos.dat`. Cada registro de dicho fichero está formado por dos campos: el código de cliente (valor entero tipo `long`) y la información del movimiento (cadena de 112 caracteres). El número de registros del fichero es desconocido.

Además, todos los códigos de clientes válidos, se encuentran almacenados en un fichero de texto `cod_cli.txt`, donde cada registro contiene un único campo que es el código del cliente. El número total de registros es desconocido pero menor que 100. Los códigos de cliente son números enteros y los registros están separados entre si por un caracter `'\n'`.

Escribir un programa completo que depure el fichero de movimientos de cuentas de clientes comprobando que el código del cliente es correcto. Los movimientos correctos se grabarán en un fichero y los erróneos en otro.

Proceso:

- Cargar el fichero de códigos de clientes (`Cod_cli.txt`, tipo texto) en un vector (estático) cuya longitud máxima será 100.
- Procesar el fichero de movimientos (`Movimientos.dat`) en su totalidad escribiendo un fichero con los movimientos cuyos códigos de cliente son correctos (`MovsBuenos.dat`) y otro fichero con los movimientos cuyos códigos de cliente son incorrectos (`MovsMalos.dat`).
- Como información final mostrar por pantalla el número total de movimientos, el nº. de movimientos "buenos" y el nº. de movimientos "malos".

Usar la siguiente función para la búsqueda del código del cliente:

```
int BuscarCliente(long cod_cli[], int num_clientes, long cod_a_buscar);
```

Esta función devolver cero (0) si no se encuentra el cliente y uno (1) si el cliente es correcto.

Usar las restantes funciones que el alumno considere oportunas.