UNIVERSIDAD PONTIFICIA COMILLAS
Escuela Técnica Superior de Ingeniería (ICAI)
Ingeniero en Informática
Programación Básica

27 de junio de 2003 Alumno/a:_____Grupo: | A | B |

Apartado 1. (**1 punto**). Dado el siguiente programa, indicar cuál será la salida que produce:

```
#include <stdio.h>

void cambio1(int *, int *);
void cambio2(double *, double *);

void main()
{
        int    a, b;
        int    *pa, *pb;

        double x, y;
        double *px, *py;

        a=5;
        b=7;

        x=3.141592;
        y=2.718281;

        pa=&a;
        pb=&b;

        px=&x;
        py=&y;

        cambio1(&a, &b);
        printf("\nValores 1: %d %d", a, b);

        cambio2(&x, &y);
        printf("\nValores 2: %f %f", x, y);

        cambio1(pa,pb);
        printf("\nValores 3: %d %d", a, b);

        cambio2(px, py);
        printf("\nValores 4: %f %f", x, y);

        printf("\n\nFin de programa\n\n");
}

void cambio1(int *a, int *b)
{
        int *c;
        c=a;
        a=b;
        b=c;
}

void cambio2(double *x, double *y)
{
        double *z;
        z=x;
        x=y;
        y=z;
}
```

Nota: En este examen no se pueden utiulizar las sentencias `break`, `continue` y `exit`. En cada función sólo puede haber una sentencia `return` al final de la misma.

Valores 1: 5 7

Valores 2: 3.141592 2.718281

Valores 3: 5 7

Valores 4: 3.141592 2.718281

Fin de programa

Apartado 2. (**2 puntos**). Codificar una función `Capicua` que reciba un número entero, positivo e inferior a 100000 y devuelva el valor -1 si el número está fuera de rango, 1 si es capicua y 0 si no lo es.

```
/* Funcion:        Capicua
   Descripcion:    Recibe un entero y devuelve 1 si es capicua, 0 si no lo es y -1 está fuera de rango. */
int Capicua (int n)
{
    int unidades, decenas, centenas, unidad_millar, decena_millar;
    int valor = 0;

    if (n < 0 || n > 99999)                          // Control  fuera de rango
        valor = -1;
    else
    {
        if (n < 10)                                  // Numero de 1 cifra
            valor = 1;
        else  if (n < 100)                           // Numero de 2 cifras
            {
                unidades = n % 10;
                decenas = n / 10;
                if (unidades == decenas)
                    valor = 1;
            }
            else  if (n < 1000)                      // Numero de 3 cifras
                {
                    unidades = n % 10;
                    centenas = n / 100;
                    if (unidades == centenas)
                        valor = 1;
                }
                else  if (n < 10000)                 // Numero de 4 cifras
                    {
                        unidades = n % 10;
                        decenas = (n / 10) % 10;
                        centenas = (n / 100) % 10;
                        unidad_millar = n / 1000;
                        if (unidades == unidad_millar && decenas == centenas)
                            valor = 1;
                    }
                    else                             // Numero de 5 cifras
                    {
                        unidades = n % 10;
                        decenas = (n / 10) % 10;
                        unidad_millar = (n / 1000) % 10;
                        decena_millar = n / 10000;
                        if (unidades == decena_millar && decenas == unidad_millar)
                            valor = 1;
                    }
    }
    return valor;
}
```

Apartado 3. (**1 punto**). Suponiendo que las variables **a** y **c** del siguiente programa están almacenadas respectivamente en las direcciones de memoria **3AF8** y **5A62**, ***rellenar la siguiente tabla*** indicando los valores que van tomando todas las variables y punteros del programa, después de la ejecución de cada línea.
Las variables que no sean de interés se pueden marcar con guiones (---), y si los valores son desconocidos se pueden marcar con el signo de interrogación (?) (ver ejemplo en línea 0). ***Indicar además la salida del programa.***

```
#include <stdio.h>
int Func1(int x, int y);
void Func2(int *m, int *n);

void main(void)
{
    int a;
    int b;
    int c;
                                /* (0) */
    a=4;                        /* (1) */
    b=3;                        /* (2) */
    c=Func1(a,b);               /* (3) */
    Func2(&c,&a);               /* (4) */
    printf("%d\n",a);           /* (5) */
}


int Func1(int x, int y)
{
    int i;                      /* (10) */

    for(i=0; i<y; i++)
    {                           /* (11) */
        x *= y;                 /* (12) */
    }
    return(x);                  /* (13) */
}

void Func2(int *m, int *n)
{                               /* (20) */
    *n = *m;                    /* (21) */
}
```

	a	b	c	x	y	i	m	n
0	?	?	?	---	---	---	---	---
1	**4**	**?**	**?**	---	---	---	---	---
2	**4**	**3**	**?**	---	---	---	---	---
3	**4**	**3**	**108**	---	---	---	---	---
4	**108**	**3**	**108**	---	---	---	---	---
5	**108**	**3**	**108**	---	---	---	---	---
10	---	---	---	**4**	**3**	**?**	---	---
11	---	---	---	**4**	**3**	**0**	---	---
12	---	---	---	**12**	**3**	**0**	---	---
11	---	---	---	**12**	**3**	**1**	---	---
12	---	---	---	**36**	**3**	**1**	---	---
11	---	---	---	**36**	**3**	**2**	---	---
12	---	---	---	**108**	**3**	**2**	---	---
13	---	---	---	**108**	**3**	**3**	---	---
20	**4**	---	**108**	---	---	---	**5A62**	**3AF8**
21	**108**	---	**108**	---	---	---	**5A62**	**3AF8**

Salida del programa: **108**

Apartado 4. **(2 puntos).** El siguiente programa carga en memoria una lista simplemente encadenada con información de los cursos de Informática que se realizan en una determinada organización. La información de cada curso se compone de un título del curso, un código numérico entero de 3 cifras y el precio en euros. La parte de carga de datos en la lista y su eliminación ya se encuentran realizadas como puede verse en el siguiente código.

Se pide: ***Realizar el segmento de codigo que, una vez cargados los datos en la lista, nos permita realizar consultas sucesivas del precio de un curso a través de su código, las consultas pueden terminar al introducir un código 0.***

```
/* Gestion de una lista encadenada de
   Cursos con titulo, codigo y precio */

// Directivas del precompilador
#include <conio.h>
#include <stdio.h>
#include <string.h>

// Programa principal
void main()
{
        int i;

        // Estructura de los elementos de la lista
        typedef struct curso
        {
                char   titulo[20];
                int    codigo;
                float  precio;
                struct curso *siguiente;
        }CURSO;

        CURSO *inicio;   // Puntero al comienzo de la lista
        CURSO *actual;   // Puntero al elemento actual
        CURSO *aux;      // Puntero auxiliar

        char seguir;
        int  xcodigo;

        // Inicializacion de la lista
        inicio = NULL;
        i=0;

        printf("\n\nDesea introducir datos de un curso...(s/n): ");
        scanf("%c",&seguir);

        // Bucle de creacion de los elementos de la lista
        while (seguir=='s' || seguir=='S')
        {
                system ("cls");
                // Peticion de memoria dinámica para el nuevo curso
                actual = (CURSO *) malloc (sizeof(CURSO));

                if(i==0)
                        inicio=actual;
                else
                        aux->siguiente=actual;

                // Lectura de los datos de un curso
                i++;
                fflush(stdin);
                printf("\n\nIntroduce el titulo del curso %2d: ", i);
                gets(actual->titulo);
                printf("\nIntroduce el codigo (3 cifras):   ");
                scanf("%d", &actual->codigo);
                printf("\nIntroduce el precio:              ");
                scanf("%f", &actual->precio);
                actual->siguiente=NULL;
                aux=actual;

                printf("\n\nDesea introducir datos de algun curso mas...(s/n): ");
                fflush(stdin);
                scanf("%c",&seguir);
        }

        // Consulta de un precio por el codigo del curso
        ...
        ...
        ...
        ...
        ...
        ...



        // Eliminacion de los elementos de la lista
        actual=inicio;
        printf("\n\n*** Eliminando elementos de la lista ***\n\n");
        i=0;
        while (actual!=NULL)
        {
                i++;
                aux=actual;
                actual=actual->siguiente;
                free(aux);
                printf("\n\nEliminado elemento %d...", i);
                getch();
        }

        system ("cls");
        printf("\n\n *** FIN DEL PROGRAMA ***\n\n");
}
```

Este segmento de programa es el que se pide

```
…
// Consulta de un precio por el codigo del curso
printf ("\n\n\nBusqueda de precios de cursos por su codigo:\n\n\n");

// Lectura del primer codigo a consultar
do
{
        printf (“Introducir el codigo del curso ( 0 para terminar )...”);
        scanf (“%d”, &xcodigo);
} while (xcodigo < 0 || xcodigo > 999);

// Bucle de repeticion de consultas
while ( xcodigo != 0)
{
        // Busqueda del precio del curso
        actual = inicio;
        while (actual->siguiente != NULL && actual->codigo != xcodigo)
                actual = actual->siguiente;
        if (actual->codigo == xcodigo)
                printf ("\nEl precio del curso es de %.2f euros\n\n", actual->precio);
         else
                printf ("\nEl curso no existe en la lista\n\n");
        // Lectura del segundo y sucesivos codigos a consultar
        do
        {
               printf (“Introducir el codigo del curso ( 0 para terminar )...”);
               scanf (“%d”, &xcodigo);
        } while (xcodigo < 0 || xcodigo > 999);
}
…
```

Apartado 5. (**2 puntos**) Una biblioteca tiene un fichero binario con datos de los libros. Cada registro contiene los campos de la siguiente estructura:

```
typedef struct {
    char titulo[N];
    char autores[N];
    unsigned long isbn;  /* Código ISBN */
    int num;             /* Número de ejemplares en inventario */
    int ndisp;           /* Número de ejemplares disponibles */
} LIBRO;
```

Se quiere añadir un nuevo campo a la base de datos, para indicar el año de publicación del libro, por lo tanto la nueva estructura será:

```
typedef struct {
    char titulo[N];
    char autores[N];
    unsigned long isbn;   /* Código ISBN */
    int num;              /* Número de ejemplares en inventario */
    int ndisp;            /* Número de ejemplares disponibles */
    unsigned int editado; /* Año de edición */
} LIBRO2;
```

Para poder actualizar la base de datos actual, se nos pide un programa de conversión. Dicho programa deberá leer todos los datos de los libros uno por uno, mostrará al usuario el título, los autores y el ISBN y preguntará el año de publicación. A continuación guardará todos los datos en el fichero que maneja el nuevo formato.
Está totalmente prohibido utilizar vectores y asignación dinámica de memoria, ya que el ejercicio consiste en leer de un fichero y escribir en otro sin ninguna restricción de memoria RAM. Suponiendo que el fichero original se llama **bd_vieja** y el nuevo **bd_nueva**, completar el programa de conversión.

```
#include <stdio.h>
#include <sring.h>
#define N 50
typedef struct {
    char titulo[N];
    char autores[N];
    unsigned long isbn;  /* Código ISBN */
    int num;             /* Número de ejemplares en inventario */
    int ndisp;           /* Número de ejemplares disponibles */
} LIBRO;
typedef struct {
    char titulo[N];
    char autores[N];
    unsigned long isbn;  /* Código ISBN */
    int num;             /* Número de ejemplares en inventario */
    int ndisp;           /* Número de ejemplares disponibles */
    unsigned int editado;/* Año de edición */
} LIBRO2;
void main(void)
{
    LIBRO   viejo;       /* registro de la base de datos vieja */
    LIBRO2  nuevo;       /* registro de la base de datos nueva */
    FILE   *fp_vieja;    /* puntero para la base de datos vieja */
    FILE   *fp_nueva;    /* puntero para la base de datos nueva */
    // Apertura del fichero bd_vieja
    fp_vieja = fopen ("c:\\temp\\bd_vieja.dat", "rb");
    if (fp_vieja == NULL)
        printf ("\nError en la apertura del fichero antiguo\n);
    else
    {
```

```
        // Apertura del fichero bd_nueva
        fp_nueva = fopen ("c:\\temp\\bd_nueva.dat", "wb");
        if (fp_vieja == NULL)
            printf ("\nError en la apertura del fichero nuevo\n);
        else
        {
            // Lectura del primer registro
            fread (viejo, sizeof(LIBRO), 1, fp_vieja);
            while (!feof(fp_vieja))
            {
                // Presentacion de los datos del libro
                printf ("\nTitulo:  %s", viejo.titulo);
                printf ("\nAutores: %s", viejo.autores);
                printf ("\nISBN:    %lu", viejo.isbn);
                // Lectura del año de publicacion
                printf ("\nIntroducir el año de publicacion: ");
                scanf ("%u", &nuevo.editado);
                // Paso de datos del buffer antiguo al nuevo
                strcpy (nuevo.titulo, viejo.titulo);
                strcpy (nuevo.autores, viejo.autores);
                nuevo.isbn = viejo.isbn;
                nuevo.num = viejo.num;
                nuevo.ndisp = viejo.ndisp;
                // Escribimos el nuevo registro
                fwrite (nuevo, sizeof(LIBRO2), 1, fp_nueva);
                // Lectura del siguiente registro
                fread (viejo, sizeof(LIBRO), 1, fp_vieja);
            }
            fclose (fp_nueva);
        }
        fclose (fp_vieja);
    }
}
```

Apartado 6. (**1 punto**) Escribe ejemplos de `typedef` y de declaración de las variables `a`, `b` y `c` para que las siguientes operaciones tengan sentido:

```
a[5].longitud = 34.76;
a[2].pala[0] = '\0';
b.balde[5] = 12345;
b.punto.x = 5.34;
c->fecha.dia = 12;
c->nombre[3] = 'c';
```

```
// Declaraciones de tipos
typedef struct
 {
      double longitud;
      char   pala [10];
 } t_a;
 typedef struct
 {
      double x;
      double y;
 } t_coord;
 typedef struct
 {
      int    balde [10];
      t_coord punto;
 } t_b;
 typedef struct
 {
      int dia;
      int mes;
      int anio;
 } t_fecha;
 typedef struct
 {
      t_fecha fecha;
      char nombre [30];
 } t_c;
 // Declaraciones de variables
 t_a a[100];
 t_b b;
 t_c *c;
```

Apartado 7. (**1 punto**) Escribir una función **Duplicar** que reciba una palabra y retorne una nueva palabra duplicando cada una de las letras de la misma. La función debe calcular el número de letras de la palabra original y reservar memoria dinámica suficiente para la nueva palabra.
Por ejemplo: si la función recibe la palabra "hola", debe generar y retornar la palabra "hhoollaa".

```
/* Funcion:        Duplicar
   Descripcion:    Recibe una palabra y la devuelve con sus letras duplicadas
   En caso de error esta función devuelve NULL
*/
char * Duplicar (char * palabra)
{
      char * palabra2;
      int  longitud, longitud2;
      longitud = strlen (palabra);
      longitud2 = 2*longitud+1;
      palabra2 = (char *) calloc (longitud2, sizeof(char));
      if  (palabra2 == NULL) {
            printf("Error por falta de memoria\n");
      } else  {
            for (i = 0; i < longitud; i++)
            {
                  palabra2 [2*i] = palabra [i];
                  palabra2 [2*i + 1] = palabra [i];
            }
            palabra [longitud2] = '\0';
      }
      return palabra2;
}
```