Informática II **Febrero 2012**
FINAL **Tema 1**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: _____

PARTE I – Complete el programa (26 puntos, 13 cada ejercicio)

1.1- Completar el código de manera tal que la función booleana **es_palindromo** verifique si la palabra recibida es palíndromo o no, devolviendo VERDADERO o FALSO según corresponda.
NOTA: es obligatorio el uso de sintaxis de punteros para acceder al arreglo.

```
int es_palindromo(char *palabra)
{
  int i, ini, fin, res = 1;

  for (i = 0; *(palabra+i) != '\0'; i++);

  ini = 0;
  fin = i-1;

  while (_____________________________)
  {
    if (_____________________________)
    {
      res = 0;
    }
    _________________;
    _________________;
  }

  return _________________;
}
```

1.2- Completar el código de manera tal que la función recursiva **cantidad_niveles** devuelva la cantidad de niveles del árbol recibido como parámetro.

```
int cantidad_niveles(t_nodo_bin_ptr pNodo)
{
  int izq, der;

  if (pNodo != NULL)
  {
    izq = _____________________________;
    der = _____________________________;

    if (_____________________________)
    {
      return 1 + izq;
    }
    else
    {
      _____________________________;
    }
  }
  else
  {
      return 0;
  }
}
```

```
typedef struct s_nodo_bin
{
    int valor;
    struct s_nodo_bin* pIzq;
    struct s_nodo_bin* pDer;
} *t_nodo_bin_ptr;
```

PARTE II – Opción múltiple (26 puntos, solamente si están todas correctas)

2 - Marque la opción correcta que se corresponda con lo que el código imprimiría por pantalla:

A)
```
char a;
a= ((((1) << (1<<2)) >> 4) & 255) << 7;
printf("%d",a);
```

(A) 240 (B) -128 (C) 2 (D) 15 (E) 16 (F) -240
(G) Ninguna de las anteriores

B)
```
char *b = "12345";
char *  p = b;
int i;
for(i=0; *(p+i)!='\0';p++);
printf( "%s%s", p,b );
```

(A) 1234512345 (B) 1122334455 (C) 12345
(D) 123 (E) 54321 (F) Ninguna de las anteriores

C)
```
unsigned int i=0;

for(i=1;i<10;i=i<<1)
{
   printf("%d", i);
}
```

(A) 1111111111 (B) 0123456789 (C) 12345678910
(D) 123 (E) 1248 (F) Ninguna de las anteriores

D)
```
char x (char a)
{
    if (a<='D')
        printf("%c", x(a+1));
    else
    {
        printf("%c", a);
        return a-1;
    }

    return a;
}

void main()
{
    printf("%c",x('A'));
}
```

(A) EDDCBA (B) AAAAAA (C) ABCDE (D) EDCBA
(E) AEDCBE (F) Ninguna de las anteriores

Informática II **Febrero 2012**
FINAL **Tema 1**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: _____

PARTE III – Programación (48 puntos, 24 cada ejercicio)

Archivo	Contenido
Parcelas.txt	Registros: ID_Parcela (int) ID_Categoria (int) Tamanio (int / metros cuadrados) Valor (double)
Categorias.dat	Registros: ID_Categoria Descripcion (string) SeccionJurisdiccion (string)

1- Realizar una función CARGA_SUMA que retorne un árbol binario de búsqueda con la siguiente estructura:

```
struct s_hoja
{
   ID_Categoria    int;
   Descripcion     char[50];
   Cant_Parcelas   int;
   struct s_hoja    *    pizq;
   struct s_hoja    *    pder;
};
typedef struct s_hoja *t_arbol;
```

siendo el valor de búsqueda el ID_Categoria por lo tanto dichos identificadores no se van a ver repetidos dentro del árbol.

La función debe cargar el árbol sumando la cantidad de parcelas para cada categoría. Puede haber categorías que no tengan parcelas asociadas por lo tanto se recomienda primero cargar el árbol con todas las categorías y luego realizar la suma correspondiente.

2- Realizar una función VERIFICACION que reciba un árbol con la estructura del ejercicio anterior y que verifique si todas las categorías que se encuentran en el árbol, existen en el archivo de categorias.dat En caso de que haya una categoría en el árbol y no en el archivo, debe agregarse al final del archivo, usando los datos de ID y Descripcion que se encuentran en el nodo y poniendo como Seccion/Jurisdiccion la palabra "DESCONOCIDA." En caso de haber Categorias en el archivo que no estén en el árbol, no se debe hacer nada.

Condiciones para todos los ejercicios:

- Para aprobar el examen se debe sumar un minimo de 65 puntos.
- Declarar todos los "struct" y "typedef" que utilice para resolver el ejercicio.
- No desperdiciar memoria. Liberar todos los espacios que se hayan reservado y que no se utilicen.
- Se pueden programar funciones adicionales, respetando los puntos anteriores.
- Considere que las estructuras recibidas como parámetro por las funciones pueden estar vacías (nulas).
- Las pilas y colas deben respetar el protocolo y el método elegido para su implementación a lo largo de todo el examen.
