Informática II **Febrero 2011**
FINAL **Tema 1**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: ____

PARTE I – Complete el programa (3 puntos)

Para realizar este ejercicio debe modificar el código que se indica de manera tal que cumpla con el enunciado. Complete las secciones indicadas con "**___**" con el código correspondiente. Complete las secciones indicadas con "**__(*)__**" con una o mas líneas de código (todas las que considere necesarias.) El resto de las lineas de código del programa deben quedar sin modificar, y en el orden mostrado:

1- Completar el código de manera tal que la función **agregar** agregue un nodo al final de una lista doblemente enlazada de números enteros.

```
struct s_nodo
{ int valor;
  struct s_nodo* sig;
  struct s_nodo* ant;
};
typedef struct s_nodo* t_nodo;

t_nodo nuevo_nodo(int valor)
{ t_nodo aux;
  __(*)__
  aux->valor = valor;
  aux->ant   = NULL;
  aux->sig   = NULL;
  return aux;
}
void agregar(t_nodo* lista, int valor)
{ t_nodo aux;
  if (*lista == NULL)
  {
    ____;
  } else {
    aux = *lista;
    while (aux->sig != NULL)
        aux = aux-> sig;
    __(*)__
  }
}
```

2- Completar el código de manera tal que la función recursiva **eliminarSub** encuentre y elimine el subarbol completo donde se encuentra el valor del parámetro.

```
struct s_nodo_bin
{ int valor;
  struct s_nodo_bin* izq;
  struct s_nodo_bin* der;
};
typedef struct s_nodo_bin* t_nodo_bin;
void borrarNodo(t_nodo_bin* arbol)
{ if (*arbol != NULL)
  {
    borrarNodo( ___ );
    borrarNodo( ___ );
    free( ___ );
    *arbol=NULL;
  }
}
void eliminarSub(t_nodo_bin* arbol, int valor)
{ if (*arbol == NULL)
       return;
  else
  { if (valor == (*arbol)-> valor)
          borrarNodo( ___ );
    else
        if (valor < (*arbol)->valor)
              ___;
        else
              ___;
    }
    return;
}
```

PARTE II – Opción múltiple (2 puntos, solamente si están todas correctas)

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

3)
```
int a=9, b=13, c=-3, d=8;
int *bb, **cc;
bb=&a;
cc= &bb;
(*bb)++;
a= 5;
**cc = **cc + 4;
*cc = &d;
printf("%d", *bb);
```

(A) 12 (B) 13 (C) 9 (D) 8 (E) 14 (F) 10

4)
```
char x (char a)
   {
    if (a>'w')
      printf("%c", x(a-1));
    return 'w';
   }
   void main()
   { printf("%c",x('y'));
   }
```

(A) yxw (B) wxy (C) uvww (D) wyxw
(E) wwww (F) ywxwww (G) www

5)
```
char *b = "3456789";
char *  p = b;
p += 5;
printf( "%s%s", p,b );
```

(A) 5456789 (B) 89456789 (C) 54567893456789 (D) 34567943456794
(E) 893456789 (F) 34567943456789

6)
```
unsigned char a;
a=  ( (1) << (1<<2));
printf("%d",a);
```

(A) 240 (B) 1 (C) 2 (D) 15 (E) 16
(F) -240 (G) 4

PARTE III – Programación (5 puntos)

Archivo	**Contenido**
Actores.dat	Registros: Apellido del Actor (char [50]) Nombre del actor (char [50]) Código de actor (int)
Peliculas.dat	Registros: Código de película (int) Nombre de la pelicula (char [80]) Año de estreno (int)
Peliculas_Actores.txt	Archivo CSV con: Código de película (numérico) Código de actor (numérico)

7) Realizar una función que reciba como parámetro dos números enteros que representan dos códigos de actor y que imprima por pantalla un listado (incluyendo nombre y año de estreno) de todas las películas donde ambos actores hayan trabajado juntos.

8) Realizar una función que reciba como parámetros el NOMBRE de una película; y que imprima por pantalla todos los nombres y apellidos de los actores que hayan actuado en la misma. Puede asumir que el nombre de película es único.

- Declarar todos los "struct" y "typedef" que utilice para resolver estos ejercicios.
- No desperdiciar memoria. Liberar todos los espacios que se hayan reservado y que no se utilicen.
- En los ejercicios de programación se pueden programar funciones adicionales a las pedidas.

Informática II **Febrero 2011**
FINAL **Tema 2**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: ____

PARTE I – Complete el programa (3 puntos)

Para realizar este ejercicio debe modificar el código que se indica de manera tal que cumpla con el enunciado. Complete las secciones indicadas con "**___**" con el código correspondiente. Complete las secciones indicadas con "**__(*)__**" con una o mas líneas de código (todas las que considere necesarias.) El resto de las lineas de código del programa deben quedar sin modificar, y en el orden mostrado:

1- Completar el código de manera tal que la función recursiva **eliminarSub** encuentre y elimine el subarbol completo donde se encuentra el valor del parámetro.

```
struct s_nodo_bin
{ int valor;
  struct s_nodo_bin* izq;
  struct s_nodo_bin* der;
};
typedef struct s_nodo_bin* t_nodo_bin;
void borrarNodo(t_nodo_bin* arbol)
{ if (*arbol != NULL)
  {
    borrarNodo( ___ );
    borrarNodo( ___ );
    free( ___ );
    *arbol=NULL;
  }
}
void eliminarSub(t_nodo_bin* arbol, int valor)
{ if (*arbol == NULL)
       return;
  else
  { if (valor == (*arbol)-> valor)
         borrarNodo( ___ );
    else
        if (valor < (*arbol)->valor)
             ___;
        else
             ___;
    }
    return;
}
```

2- Completar el código de manera tal que la función **agregar** agregue un nodo al final de una lista doblemente enlazada de números enteros.

```
struct s_nodo
{ int valor;
  struct s_nodo* sig;
  struct s_nodo* ant;
};
typedef struct s_nodo* t_nodo;

t_nodo nuevo_nodo(int valor)
{ t_nodo aux;
  __(*)__
  aux->valor = valor;
  aux->ant   = NULL;
  aux->sig   = NULL;
  return aux;
}
void agregar(t_nodo* lista, int valor)
{ t_nodo aux;
  if (*lista == NULL)
  {
    ____;
  } else {
    aux = *lista;
    while (aux->sig != NULL)
        aux = aux-> sig;
    __(*)__
  }
}
```

PARTE II – Opción múltiple (2 puntos, solamente si están todas correctas)

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

3)
```
unsigned char a;
a=  ( (1) << (1<<2));
printf("%d",a);
```

(A) 240 (B) 16 (C) 2 (D) 15 (E) 1 (F) 7 (G) 4

4)
```
char x (char a)
{
 if (a>'w')
   printf("%c", x(a-1));
 return 'w';
}
void main()
{ printf("%c",x('y'));
}
```

(A) yxw (B) wxy (C) uvww (D) wyxw
(E) www (F) ywxwww (G) wwww

5)
```
char *b = "3456789";
char *  p = b;
p += 5;
printf( "%s%s", p,b );
```

(A) 5456789 (B) 89456789 (C) 54567893456789
(D) 34567943456794 (E) 893456789 (F) 34567943456789

6)
```
int a=9, b=13, c=-3, d=8;
int *bb, **cc;
bb=&a;
cc= &bb;
(*bb)++;
**cc = **cc + 4;
*cc = &d;
printf("%d", *bb);
```

(A) 12 (B) 13 (C) 9 (D) 8 (E) 14 (F) 10

PARTE III – Programación (5 puntos)

Archivo	**Contenido**
Actores.dat	Registros: Apellido del Actor (char [50]) Nombre del actor (char [50]) Código de actor (int)
Peliculas.dat	Registros: Código de película (int) Nombre de la pelicula (char [80]) Año de estreno (int)
Peliculas_Actores.txt	Archivo CSV con: Código de película (numérico) Código de actor (numérico)

7) Realizar una función que reciba como parámetro dos números enteros que representan dos códigos de actor y que imprima por pantalla un listado (incluyendo nombre y año de estreno) de todas las películas donde ambos actores hayan trabajado juntos.

8) Realizar una función que reciba como parámetros el NOMBRE de una película; y que imprima por pantalla todos los nombres y apellidos de los actores que hayan actuado en la misma. Puede asumir que el nombre de película es único.

- Declarar todos los "struct" y "typedef" que utilice para resolver estos ejercicios.
- No desperdiciar memoria. Liberar todos los espacios que se hayan reservado y que no se utilicen.
- En los ejercicios de programación se pueden programar funciones adicionales a las pedidas.

Informática II **Febrero 2011**
FINAL **Tema 3**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: ____

PARTE I – Complete el programa (3 puntos)

Para realizar este ejercicio debe modificar el código que se indica de manera tal que cumpla con el enunciado. Complete las secciones indicadas con "**___**" con el código correspondiente. Complete las secciones indicadas con "**__(*)__**" con una o mas líneas de código (todas las que considere necesarias.) El resto de las lineas de código del programa deben quedar sin modificar, y en el orden mostrado:

1- Completar el código de manera tal que la función **agregar** agregue un nodo al final de una lista doblemente enlazada de números enteros.

```
struct s_nodo
{ int valor;
  struct s_nodo* sig;
  struct s_nodo* ant;
};
typedef struct s_nodo* t_nodo;

t_nodo nuevo_nodo(int valor)
{ t_nodo aux;
  __(*)__
  aux->valor = valor;
  aux->ant   = NULL;
  aux->sig   = NULL;
  return aux;
}
void agregar(t_nodo* lista, int valor)
{ t_nodo aux;
  if (*lista == NULL)
  {
    ____;
  } else {
    aux = *lista;
    while (aux->sig != NULL)
        aux = aux-> sig;
    __(*)__
  }
}
```

2- Completar el código de manera tal que la función recursiva **eliminarSub** encuentre y elimine el subarbol completo donde se encuentra el valor del parámetro.

```
struct s_nodo_bin
{ int valor;
  struct s_nodo_bin* izq;
  struct s_nodo_bin* der;
};
typedef struct s_nodo_bin* t_nodo_bin;
void borrarNodo(t_nodo_bin* arbol)
{ if (*arbol != NULL)
  {
    borrarNodo( ___ );
    borrarNodo( ___ );
    free( ___ );
    *arbol=NULL;
  }
}
void eliminarSub(t_nodo_bin* arbol, int valor)
{ if (*arbol == NULL)
       return;
  else
  { if (valor == (*arbol)-> valor)
          borrarNodo( ___ );
    else
        if (valor < (*arbol)->valor)
              ___;
        else
              ___;
    }
    return;
}
```

PARTE II – Opción múltiple (2 puntos, solamente si están todas correctas)

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

3)
```
int a=9, b=13, c=-3, d=8;
int *bb, **cc;
bb=&a;
cc= &bb;
(*bb)++;
a= 5;
**cc = **cc + 4;
*cc = &d;
printf("%d", *bb);
```

(A) 12 (B) 13 (C) 9 (D) 8 (E) 14 (F) 10

4)
```
char x (char a)
  {
   if (a>'w')
     printf("%c", x(a-1));
   return 'w';
  }
  void main()
  { printf("%c",x('y'));
  }
```

(A) yxw (B) wxy (C) uvww (D) wyxw
(E) wwww (F) ywxwww (G) www

5)
```
char *b = "3456789";
char *  p = b;
p += 5;
printf( "%s%s", p,b );
```

(A) 5456789 (B) 89456789 (C) 54567893456789 (D) 34567943456794
(E) 893456789 (F) 34567943456789

6)
```
unsigned char a;
a=  ( (1) << (1<<2));
printf("%d",a);
```

(A) 240 (B) 1 (C) 2 (D) 15 (E) 16 (F) -240 (G) 4

PARTE III – Programación (5 puntos)

Archivo	**Contenido**
Actores.dat	Registros: Apellido del Actor (char [50]) Nombre del actor (char [50]) Código de actor (int)
Peliculas.dat	Registros: Código de película (int) Nombre de la pelicula (char [80]) Año de estreno (int)
Peliculas_Actores.txt	Archivo CSV con: Código de película (numérico) Código de actor (numérico)

7) Realizar una función que reciba como parámetro dos números enteros que representan dos códigos de actor y que imprima por pantalla un listado (incluyendo nombre y año de estreno) de todas las películas donde ambos actores hayan trabajado juntos.

8) Realizar una función que reciba como parámetros el NOMBRE de una película; y que imprima por pantalla todos los nombres y apellidos de los actores que hayan actuado en la misma. Puede asumir que el nombre de película es único.

- Declarar todos los "struct" y "typedef" que utilice para resolver estos ejercicios.
- No desperdiciar memoria. Liberar todos los espacios que se hayan reservado y que no se utilicen.
- En los ejercicios de programación se pueden programar funciones adicionales a las pedidas.

Informática II **Febrero 2011**
FINAL **Tema 4**

APELLIDO Y NOMBRE:_____ CANTIDAD DE HOJAS: ____

PARTE I – Complete el programa (3 puntos)

Para realizar este ejercicio debe modificar el código que se indica de manera tal que cumpla con el enunciado. Complete las secciones indicadas con "**___**" con el código correspondiente. Complete las secciones indicadas con "**__(*)__**" con una o mas líneas de código (todas las que considere necesarias.) El resto de las lineas de código del programa deben quedar sin modificar, y en el orden mostrado:

1- Completar el código de manera tal que la función recursiva **eliminarSub** encuentre y elimine el subarbol completo donde se encuentra el valor del parámetro.

```
struct s_nodo_bin
{ int valor;
  struct s_nodo_bin* izq;
  struct s_nodo_bin* der;
};
typedef struct s_nodo_bin* t_nodo_bin;
void borrarNodo(t_nodo_bin* arbol)
{ if (*arbol != NULL)
  {
    borrarNodo( ___ );
    borrarNodo( ___ );
    free( ___ );
    *arbol=NULL;
  }
}
void eliminarSub(t_nodo_bin* arbol, int valor)
{ if (*arbol == NULL)
       return;
  else
  { if (valor == (*arbol)-> valor)
         borrarNodo( ___ );
    else
        if (valor < (*arbol)->valor)
             ___;
        else
             ___;
    }
    return;
}
```

2- Completar el código de manera tal que la función **agregar** agregue un nodo al final de una lista doblemente enlazada de números enteros.

```
struct s_nodo
{ int valor;
  struct s_nodo* sig;
  struct s_nodo* ant;
};
typedef struct s_nodo* t_nodo;

t_nodo nuevo_nodo(int valor)
{ t_nodo aux;
  __(*)__
  aux->valor = valor;
  aux->ant   = NULL;
  aux->sig   = NULL;
  return aux;
}
void agregar(t_nodo* lista, int valor)
{ t_nodo aux;
  if (*lista == NULL)
  {
    ____;
  } else {
    aux = *lista;
    while (aux->sig != NULL)
        aux = aux-> sig;
    __(*)__
  }
}
```

PARTE II – Opción múltiple (2 puntos, solamente si están todas correctas)

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

3)
```
unsigned char a;
a=  ( (1) << (1<<2));
printf("%d",a);
```

(A) 240 (B) 16 (C) 2 (D) 15 (E) 1 (F) 7 (G) 4

4)
```
char x (char a)
{
 if (a>'w')
   printf("%c", x(a-1));
 return 'w';
}
void main()
{ printf("%c",x('y'));
}
```

(A) yxw (B) wxy (C) uvww (D) wyxw
(E) www (F) ywxwww (G) wwww

5)
```
char *b = "3456789";
char *  p = b;
p += 5;
printf( "%s%s", p,b );
```

(A) 5456789 (B) 89456789 (C) 54567893456789
(D) 34567943456794 (E) 893456789 (F) 34567943456789

6)
```
int a=9, b=13, c=-3, d=8;
int *bb, **cc;
bb=&a;
cc= &bb;
(*bb)++;
**cc = **cc + 4;
*cc = &d;
printf("%d", *bb);
```

(A) 12 (B) 13 (C) 9 (D) 8 (E) 14 (F) 10

PARTE III – Programación (5 puntos)

Archivo	**Contenido**
Actores.dat	Registros: Apellido del Actor (char [50]) Nombre del actor (char [50]) Código de actor (int)
Peliculas.dat	Registros: Código de película (int) Nombre de la pelicula (char [80]) Año de estreno (int)
Peliculas_Actores.txt	Archivo CSV con: Código de película (numérico) Código de actor (numérico)

7) Realizar una función que reciba como parámetro dos números enteros que representan dos códigos de actor y que imprima por pantalla un listado (incluyendo nombre y año de estreno) de todas las películas donde ambos actores hayan trabajado juntos.

8) Realizar una función que reciba como parámetros el NOMBRE de una película; y que imprima por pantalla todos los nombres y apellidos de los actores que hayan actuado en la misma. Puede asumir que el nombre de película es único.

- Declarar todos los "struct" y "typedef" que utilice para resolver estos ejercicios.
- No desperdiciar memoria. Liberar todos los espacios que se hayan reservado y que no se utilicen.
- En los ejercicios de programación se pueden programar funciones adicionales a las pedidas.