

## EXAMEN Estructura de Datos y Algoritmos.

Febrero 2011

El examen suma 100 puntos. Para aprobar se necesitan 60.

### Ejercicio 1. (35 puntos)

Considere la siguiente función en C:

```
bool F (int *A, unsigned int n)
{
    bool res = true;
    int i, j;
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            if (i<j)
                res = res && (A[j]<A[i]);
    return res;
}
```

- ¿Qué calcula/retorna (conceptualmente) la función F, dado un arreglo de enteros de tamaño n?
- Calcule el orden (O) de tiempo de ejecución para el peor caso de la función F. El cálculo puede ser realizado esquemáticamente sobre el código mismo.
- El problema que resuelve F, ¿podría resolverse en un menor orden de tiempo de ejecución en el peor caso?. Justifique en caso negativo y en caso afirmativo escriba la función, indicando el orden (O).

### Ejercicio 2 (25 puntos)

Considere la siguiente definición del tipo LISTA, de listas de enteros en memoria dinámica:

```
struct nodoLista {
    int info;
    nodoLista *sig;
};
typedef nodoLista *LISTA;
```

Se pide, sin usar funciones o procedimientos auxiliares, implementar la siguiente función iterativa:

**InvParcial:** Dada una lista L de enteros de tipo LISTA, retorna una nueva lista (que no comparte registros de memoria con la lista parámetro) que contiene a todos los elementos de L, excepto al último, pero en orden inverso. Si L es una lista vacía, la función debe retornar la lista vacía.

Ejemplos:

Entradas	Resultado de <i>InvParcial</i>
L = [2,7,8,3]	[8,7,2]
L = [2]	[]
L = []	[]

### Ejercicio 3 (40 puntos)

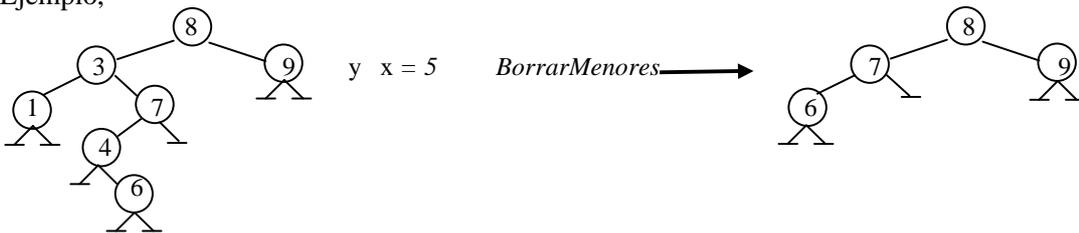
Considere la siguiente definición del tipo ABB de los árboles binarios de búsqueda de enteros, en memoria dinámica:

```
struct nodoABB {  
    int dato;  
    nodoABB * izq;    // menores que dato  
    nodoABB * der;    // mayores que dato  
};
```

```
typedef nodoABB *ABB;
```

Defina un procedimiento recursivo **BorrarMenores** en que dados un árbol binario de búsqueda de enteros A de tipo ABB y un entero x (que puede o no estar en A), elimine de A todos los elementos del árbol cuyos datos son menores a x. El resultado debe ser un árbol binario de búsqueda. El procedimiento debe evitar recorrer todo el árbol, de ser posible. Deberá liberarse la memoria de las celdas cuyos elementos sean eliminados. Se sugiere usar un procedimiento auxiliar (que deberá ser implementado) que borre todo un árbol (subárbol).

Ejemplo,



Si x fuera igual a 10 en el ejemplo previo, el árbol quedaría vacío.