Problema 1 (13 puntos: 11+2)

a) Considere la siguiente definición de listas de enteros de memoria dinámica:

```
struct nodoLista{
     int dato;
     nodoLista * sig;
}
typedef nodolista * Lista
```

Implemente una función iterativa ***Lista interseccionInversa(Lista l1, Lista l2)*** que dadas dos listas de enteros ordenadas de menor a mayor y sin elementos repetidos, construya y retorne una nueva lista ordenada de mayor a menor que contenga todos los elementos que pertenecen a ambas. La lista resultado no deberá tener elementos repetidos ni compartir memoria con las listas parámetro, que no pueden modificarse. La función debe tener O(*n*+*m*) peor caso, con *n* y *m* los largos de las listas parámetro. No utilice funciones o procedimientos auxiliares, ni estructuras de datos adicionales como arreglos/vectores.

Por ejemplo, si las listas fueran [1,3,4,5,6,8,9] y [1,2,4,7,8,11,21], el resultado debería ser: [8,4,1].

b) Justifique brevemente el orden de tiempo de ejecución requerido para la función ***interseccionInversa*****.**

Problema 2 (17 puntos: 10+7)

Para trabajar con calificaciones de estudiantes en un curso, considere la siguiente definición del tipo ***ABB*** de árboles binarios de búsqueda, que contienen números de estudiantes y notas, ambos de tipo int.

```
struct nodoABB{
     int numero;      // número de estudiante
     int nota;        // nota del estudiante
     nodoABB * izq, * der;
}
typedef nodoABB * ABB;
```

La estructura de calificaciones de estudiantes en un curso está modelada entonces con un árbol de tipo ***ABB,*** organizado (ordenado) por los números de estudiantes, que NO se pueden repetir (son las claves).

a) Implemente un procedimiento recursivo ***void actualizar(ABB & t, int nroEst, int notaEst)*** que, dado un árbol *t* de tipo ***ABB***, un número de estudiante *nroEst* y una nota *nortaEst*, inserta el estudiante *nroEst* con nota *notaEst*, si *nroEst* no estaba en *t*. En caso contrario, reemplaza la nota asociada a *nroEst* en *t* con *notaEst*. Asuma como precondición que *notaEst* es una nota válida y que *nroEst* es mayor que cero. No defina operaciones auxiliares.

Indique el orden de tiempo de ejecución en el peor caso y en el caso promedio de ***actualizar***.

b) Implemente una función recursiva ***int cantNotas(ABB t, int notaEst)*** que, dado un árbol *t* de tipo ***ABB*** y dada una nota *notaEst*, retorna la cantidad de estudiantes en *t* que tienen nota igual a *notaEst*. Si *t* es vacío, el resultado debe ser 0. Asuma como precondición que *notaEst* es una nota válida. No defina operaciones auxiliares.

Indique el orden de tiempo de ejecución en el peor caso de ***cantNotas***.