CC3001-2: Algoritmos y Estructuras de Datos
Control 2

Prof: Nelson Baloian
Aux: José M. Saavedra R.

26 de mayo, 2010
Tiempo: 1 hora 45 minutos

1. Pregunta 1: Recorrido de Árboles (2 puntos)

Considere una árbol genérico implementado bajo la clase *Arbol*:

```
public class Arbol
{
    private NodoArbol raiz;
    .
    .
    .
    public recorrerPorNivel()
    {
    }
}
```

donde al clase *NodoArbol* se define como:

```
public class NodoArbol
{
    private int dato;
private NodoArbolBinario primerHijo;
private NodoArbolBinario hermano;
.
    .
    .
    public recorrerPorNivel()
    {
    }
}
```

En este caso, cada nodo del árbol apunta, a su primer hijo y al su nodo hermano.

En un **recorrido por niveles** los nodos son visitados bajo el esquema *top-down-left-to-right*. Es decir primero son visitados los nodos del nivel k antes de visitar los nodos del nivel $k+1$. Además, en cada nivel los nodos se visitan de izquierda a derecha. **Se pide** implementar la función *recorrerPorNivel*.

Para la implementación de reccorrerPorNivel, utilze una cola que inicialmente tiene el nodo raíz, mientras la cola contenga nodos extraiga el nodo y agregue los nodos hijos apropiadamente a la cola. Los nodos se van visitando según se extraigan de la cola. (Considere que existe la clase cola con las operaciones *enqueue* y *dequeue*).

2. Pregunta 2: Árboles Binarios (2 puntos)

Dado un Árbol Binario definido como:

```
public class ArbolBinario
{
    private NodoArbolBinario raiz;
    .
    .
    boolean esABB()
    {
    }
}


public class NodoArbolBinario
{
    private int dato;
    private NodoArbolBinario nodo_Izq;
    private NodoArbolBinario nodo_Der;
    .
    .
    boolean esABB()
    {
    }
}
```

Se pide:

1. Implemente la función *esABB* que determina si un árbol binario es un árbol binario de búsqueda.
2. Implemente la misma función pero, con la restricción que ahora tome tiempo $O(n)$ (n es el tamaño del árbol).

3. Pregunta 3: Listas Enlazadas

Dada una lista enlazada, cuyos nodos almacenan valores reales. Considere que la lista se mantiene ordenada ascendentemente según el dato almacenado en los nodos. Implemente la función *reducir*, cuyo funcionamiento es como sigue: La función *reducir*, extrae los dos primero nodos del frente de la lista (i.e. los de menor valor) e inserta otro nodo cuyo dato es la suma de los datos de los dos previamente extraídos (recuerde que la lista debe seguir ordenada).

Considere un proceso P, que realiza (n-1) llamadas a *reducir* sobre una lista L con n nodos, L quedará con un solo nodo. Se pide implementar la función *reducir*.

1. Indique la complejida de *reducir*. Cúal es la complejidad del proceso P?
2. Implemente *reducir* de modo que el proceso P tenga una complejidad $O(n)$. (Hint: $dato_i + dato_{i+1} < dato_{i+2} + dato_{i+3}$ dado el orden de la lista).