Tarea No.5 de Fundamentos de programación
Ejercicios con Árboles

Ángela Villota Gómez

Universidad del Valle

Septiembre 14 de 2007

Antes de empezar:

- Esta tarea debe ser resuelta en parejas.
- En caso de copia, los grupos involucrados tienen cero.
- Debe seguir en orden los pasos de la *Estrategia de diseño*.
- Cada uno de los ejercicios debe tener su contrato, propósito, ejemplos (mínimo 3), programa y pruebas.
- el lenguaje de DrScheme debe ser: **Estudiante principiante con abreviaciones de listas**.

Importante:

Toda función que use datos compuestos debe tener una definición de los datos

Las funciones que tengan condicionales deben llevar análisis de las condiciones

Ejercicios

1. Sea $nodos_{izquierdo}$ y $nodos_{derecho}$ la cantidad de nodos que existen en el subárbol izquierdo y derecho, respectivamente, de un árbol binario. Se dice que un árbol binario está balanceado si se cumple que:

   $$|nodos_{izquierdo} - nodos_{derecho}| \leq 1$$

   Diseñe e implemente en scheme la función *balanceado?* que recibe como parámetro un árbol binario y retorna *true* si el arbol es balanceado y *false* en el caso contrario.
   *Nota:*La cantidad de nodos de un árbol binario puede definirse como la cantidad de nodos del subárbol izquierdo, más la cantidad de nodos del subárbol derecho.

2. Diseñe e implemente en scheme la función *lista_a_ABB* que recibe como argumento de entrada una lista de números enteros y retorne un árbol binario de búsqueda asociado a la lista. Ejemplo:
   si la lista es: [3 4 2 6 5 9] el árbol binario de búsqueda asociado esta en la figura No. 1.

3. Implemente en scheme las funciones *Recorrido_inorden, Recorrido_preorden* y *Recorrido_postorden*, en donde cada función recibe como argumento de entrada un árbol binario de búsqueda y retorna una lista que contenga los números enteros de cada uno de los nodos visitados en el orden en que los nodos fueron visitados.

4. Diseñe e implemente la función *contar_inorden, contar_preorden* y *contar_postorden* que recibe como argumento de entrada un árbol binario de búsqueda y retorna la cantidad de nodos visitados al ejecutar las funciones *Recorrido_inorden, Recorrido_preorden* y *Recorrido_postorden,* respectivamente.

5. Utilizando *lista_a_ABB* y una de las funciones para recorrer árboles anteriormente implementadas (inorden, preorden y postorden), diseñe e implemente la función *OrdenarABB_menor_mayor* que recibe como argumento de entrada una lista de números enteros y retorna una lista ordenada de menor a mayor.

6. Utilizando *lista_a_ABB* y una funcion para recorrer árboles (puede ser una de las anteriormente implementadas, o una nueva propuesta por usted), diseñe e implemente la función *OrdenarABB_mayor_menor* que recibe como argumento de entrada una lista de números enteros y retorna una lista ordenada de mayor a menor.

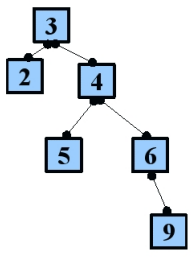

Figura 1: Arbol binario de búsqueda para el ejemplo.