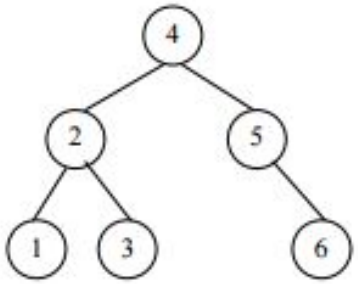
PROGRAMACION II - GUIA DE EJERCICIOS DE ÁRBOLES

Ejercicio 1:

Utilizando la estructura de un nodo para crear un Árbol Binario. Crear un programa en donde se cargue un árbol binario con valores similar al presentado en clase. Utilice un estructura de nodo binario para poder cargar el árbol de la siguiente figura. Recuerde que un árbol siempre tiene un solo nodo “Raíz” (Root) y es a partir de ese nodo desde donde se comienza a crear el resto de los nodos hijos.

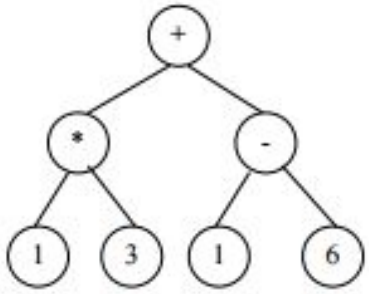


Ejercicio 2:

Utilizando el Árbol Binario del ejercicio anterior. Crear un programa que escriba todos los nodos de un árbol binario por niveles a partir del nivel 0, usando como ejemplo el árbol de la figura, siendo el resultado: **4 2 5 1 3 6**.

Ejercicio 3:

Un árbol binario puede utilizarse para almacenar una expresión algebraica, siendo las hojas los operandos y los demás nodos los correspondientes operadores.
Crear un programa para cargar un árbol binario con los valores de la siguiente figura y luego incluya un procedimiento recursivo que implemente uno de los tipos de recorridos sobre árboles (pre-orden, in-orden o post-orden) que permita obtener el resultado de la expresión en notación prefija **+ * 1 3 - 1 6**.



Ejercicio 4:

Utilizando la estructura de un nodo para crear un Árbol Binario, crear una clase BArbol (que utilice dicho Nodo internamente, similar a listas, pilas y colas) con los siguientes métodos:

BARBOL
- NodoArbol primero // define un nodo árbol inicial
+ Barbol(NodoArbol raiz) // permite asignarle una árbol a partir de su nodo raiz. **+ int getNumElementos()** // devuelve el número de elementos del árbol. **+ int getProfundidad()** // devuelve la profundidad del árbol. **+ Boolean esHoja()** // devuelve VERDADERO si el árbol es un nodo hoja, FALSO en caso contrario. **+ int caminoMinimo()** // devuelve el valor del camino con valor mínimo en el árbol. Recuerde que el valor de un camino es la suma de los valores de sus nodos. **+ BArbol copia()** // devuelve una copia del árbol **+ Boolean sonIguales(BARBOL s)** // devuelve VERDADERO si son iguales, FALSO en caso contrario. **+ Object MaxNodo()** // devuelve el mayor valor de los nodos del árbol. **+ Object MinNodo()** // devuelve el menor valor de los nodos del árbol. **+ Object MaxHoja()** // devuelve el mayor valor de las hojas del árbol. **+ Object MinHoja()** // devuelve el menor valor de los hojas del árbol. **+ Object MaxDelNivel(int nivel)** // devuelve el mayor valor de las nodos de un nivel dado del árbol. **+ Object MinDelNivel(int nivel)** // devuelve el menor valor de las nodos de un nivel dado del árbol. **+ Boolean estaEn(Object e)** // devuelve VERDADERO si el valor de “e” coincide con el de algún nodo del árbol, FALSO en caso contrario. **+ toString()** // imprime los valores del árbol en pre-orden.

Ejercicio 5:

Sabiendo que los árboles N-arios son aquellos que poseen N cantidad de hijos. Diseñe la estructura de una clase “NodoNario” para generar el árbol N-ario de la siguiente figura. Recuerde que puede utilizar para el mismo los TADs conocidos.