

# Algoritmos y Estructuras de Datos.

## 2do Parcial. Tema: **1a.** [3 de junio de 2003]

[Ej. 1] [Primitivas (10 puntos)] Escribir las funciones del TAD ARBOL ORDENADO ORIENTADO listadas a continuación, con *celdas enlazadas por punteros ó cursores*, a saber: PADRE( $n, A$ ), HIJO\_MAS\_IZQ( $n, A$ ), HERMANO\_DER( $n, A$ ), ETIQUETA( $n, A$ ), CREA2( $v, A1, A2$ ) y ANULA( $A$ ).

**Escribir todos los tipos, definiciones, funciones y procedimientos auxiliares necesarios.**

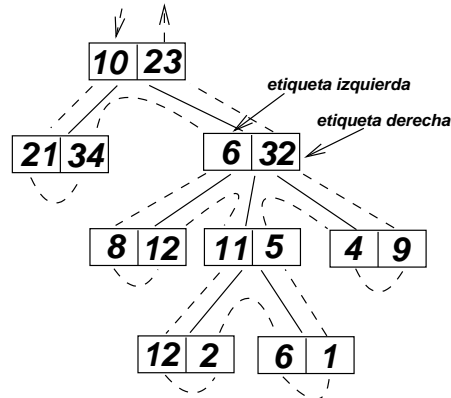
[Ej. 2] [Programación (total = 60 puntos)]

(a) [Bilistado AOO (30 puntos)]

El “bi-listado” de un árbol ordenado orientado puede pensarse como una combinación de los listados previo y posterior. Asumamos que las etiquetas tienen dos partes, una etiqueta “derecha” y una “izquierda”,

```
type tipo_etiqueta = record
    derecha, izquierda: integer
end;
```

entonces el bi-listado de un nodo  $n$  se define como la lista vacía si el nodo es  $\Lambda$  y, si no, recursivamente como la etiqueta izquierda de  $n$ , seguida del bi-listado de los hijos, seguido de la etiqueta derecha de  $n$ . Por ejemplo, para el árbol de la figura

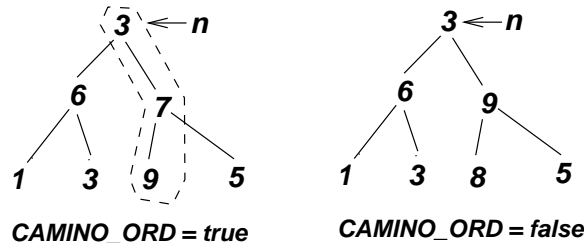


el bi-listado sigue la línea de puntos y resulta en la siguiente lista:

BI-LISTADO = {10, 21, 34, 6, 8, 12, 11, 12, 2, 6, 1, 5, 4, 9, 32, 23}. Notar que si consideramos sólo las etiquetas izuierdas, entonces el resultado es el *orden previo*, mientras que si consideramos sólo las derechas obtenemos el *orden posterior*. *Consigna:* Escribir un procedimiento `procedure BI_LISTADO(n: nodo; A: arbol);` que imprime el bi-listado de un nodo  $n$  en un árbol  $A$ . Usar las funciones del TAD ARBOL ORDENADO ORIENTADO: PADRE( $n, A$ ), HIJO\_MAS\_IZQ( $n, A$ ), HERMANO\_DER( $n, A$ ), ETIQUETA( $n, A$ ).

(b) [Camino ordenado AB (30 puntos)] Dado un árbol ordenado orientado  $A$  y un nodo  $n$  en él escribir una función `function CAMINO_ORD(n: nodo, A: arbol): boolean;` que retorna verdadero si existe algún camino ordenado desde  $n$  a una hoja en el subárbol del nodo  $n$ . Por ejemplo en el caso del árbol de la izquierda en la figura de abajo debe retornar `true` ya que el

camino marcado con líneas de puntos esta ordenado de menor a mayor yendo desde la raíz a la hoja, mientras que para el de la derecha debe retornar **false** ya que no existe ningún camino con esas características. Se sugiere el siguiente algoritmo: Un dado nodo **n** debe retornar **true** si, o bien es una hoja, o bien alguno de sus hijos **c** contiene un camino ordenado y  $ETIQUETA(c) > ETIQUETA(n)$ .



[Ej. 3] [Operativos (total = 30 puntos)]

- (a) [Árboles de Huffman (10 pts)] Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra MAREJADA.  $P(M) = 0.4, P(A) = 0.4, P(R) = 0.1, P(J) = 0.025, P(D) = 0.025, P(Z) = 0.025, P(W) = 0.025$ . Calcular la longitud promedio del código obtenido.
- (b) [Reconstruir árbol (10 pts)] Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
- $ORD\_PRE = \{Z, Q, W, P, R, M, A, B, D\}$ ,
  - $ORD\_POST = \{P, R, W, Q, A, D, B, M, Z\}$ .
- (c) [Particionar árbol (10 pts)] Considerando el árbol de la figura, decir cuál son los nodos descendientes DESC, antecesores A, izquierda I y derecha DER del nodo  $Q$ .

