

Algoritmos y Estructuras de Datos.

2do Parcial. Tema: **2b.** [3 de junio de 2003]

[Ej. 1] [Primitivas (10 puntos)] Escribir las funciones del TAD ARBOL BINARIO listadas a continuación, con *celdas enlazadas por punteros ó cursores*, a saber: PADRE(n, A), HIJO_IZQ(n, A), HIJO_DER(n, A), ETIQUETA(n, A), CREA2($v, A1, A2$) y ANULA(A). Escribir todos los tipos, definiciones, funciones y procedimientos auxiliares necesarios.

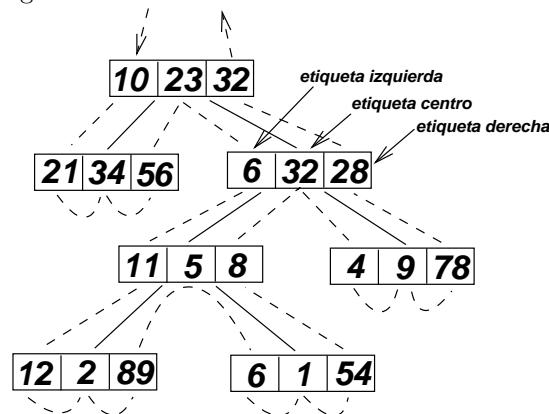
[Ej. 2] [Programación (total = 60 puntos)]

(a) [Trilistado AB (30 puntos)]

El “*tri-listado*” de un árbol binario puede pensarse como una combinación de los listados previo, posterior y simétrico. Asumamos que las etiquetas tienen tres partes “*derecha*”, “*centro*” e “*izquierda*”,

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

entonces el tri-listado de un nodo n se define como la lista vacía si el nodo es Λ y, si no, recursivamente como la etiqueta izquierda de n , seguida del tri-listado del hijo izquierdo, la etiqueta central, el tri-listado del hijo derecho y finalmente la etiqueta derecha de n . Por ejemplo, para el árbol de la figura



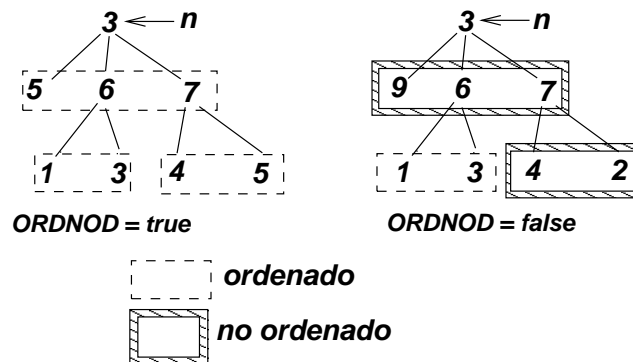
el tri-listado sigue la línea de puntos y resulta en la siguiente lista:

TRI_LISTADO = {10, 21, 34, 56, 23, 6, 11, 12, 2, 89, 5, 6, 1, 54, 8, 32, 4, 9, 78, 28, 32}. Notar que si consideramos sólo las etiquetas izquierdas, entonces el resultado es el *orden previo*, mientras que si consideramos sólo las derechas obtenemos el *orden posterior* y si consideramos las del centro, entonces obtenemos el *orden simétrico*. Consigna: Escribir un procedimiento `procedure TRI_LISTADO(n : nodo; A : arbol);` que imprime el tri-listado del subárbol de un nodo n en un árbol binario A . Usar las funciones del TAD ARBOL BINARIO: PADRE(n, A), HIJO_IZQ(n, A), HIJO_DER(n, A), ETIQUETA(n, A).

(b) [Verifica orden AOO (30 puntos)] Escribir una función

`function ORDNODE(n : nodo, A : arbol): boolean;` que verifica si cada secuencia de

hermanos del subárbol del nodo n de un árbol ordenado orientado A están ordenadas entre sí, de izquierda a derecha. Por ejemplo, para el árbol de la izquierda debería retornar **true** mientras que para el de la derecha debería retornar **false** ya que las secuencias de hermanos indicados en las *dobles cajas* rayadas NO están ordenados. Se sugiere el siguiente algoritmo. Para un dado nodo retornar **false** si: 1) sus hijos no están ordenados o 2) algunos de sus hijos contiene en su subárbol una secuencia de hermanos no ordenada (recursividad). Usar las funciones del TAD ARBOL ORDENADO ORIENTADO: PADRE(n, A), HIJO_MAS_IZQ(n, A), HERMANO_DER(n, A), ETIQUETA(n, A).



[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 EPIDEMIA. $P(E) = 0.1$, $P(P) = 0.4$, $P(I) = 0.4$, $P(D) = 0.025$, $P(M) = 0.025$, $P(A) = 0.025$, $P(W) = 0.025$. 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 = { $M, N, P, T, Q, R, S, A, B$ },
 - ORD_POST = { $N, T, R, A, B, S, Q, P, M$ }.
- (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 .

