Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Algoritmos y Estructuras de Datos

Algoritmos y Estructuras de Datos.
2do Parcial. Tema: **1b.** [3 de junio de 2003]

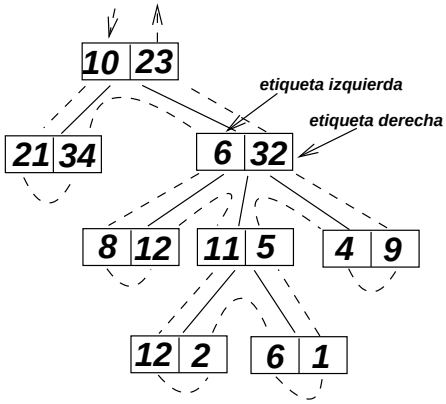
[Ej. 1] **[Primitivas (10 puntos)]** Escribir las funciones del `TAD ARBOL ORDENADO ORIENTADO` listadas a continuación, con *celdas enlazadas por punteros* **ó** *cursores*, a saber: `PADRE(n,A)`, `HIJO_MAS_IZQ(n,A)`, `HERMANO_DER(n,A)`, `ETIQUETA(n,A)`, `CREA2(v, A1, A2)` y `ANULA(A)`. **Escribir todos 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;
```

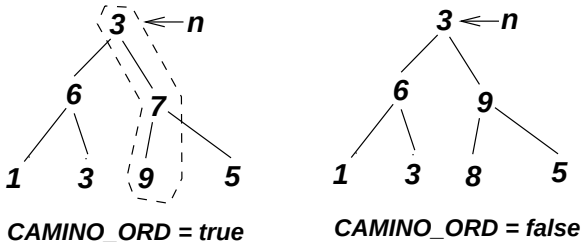
entonces el bi-listado de un nodo n se define como la lista vacía si el nodo es Λ y, si no, recursivamente como la etiqueta izquierda de n, seguida del bi-listado de los hijos, seguido de la etiqueta derecha de n. Por ejemplo, para el árbol de la figura



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 ptos)]** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra `MAREJADA`. $P(M) = 0.025, P(A) = 0.1, P(R) = 0.4, P(J) = 0.025, P(D) = 0.4, P(Z) = 0.025, P(W) = 0.025$. Calcular la longitud promedio del código obtenido.

(b) **[Reconstuir árbol (10 ptos)]** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son

- `ORD_PRE` $=\{A, W, Z, T, Q, R, S, M, N\}$,
- `ORD_POST` $=\{Z, T, W, M, N, S, R, Q, A\}$.

(c) **[Particionar árbol (10 ptos)]** Considerando el árbol de la figura, decir cuál son los nodos descendientes `DESC`, antecesores `A`, izquierda `I` y derecha `DER` del nodo Q.