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: 1A. [23 de mayo de 2006]

[Ej. 1] **[clases (20 puntos)]** Escribir la implementación en `C++` del `TAD Arbol Ordenado Orientado` (clase `tree`). Para la clase `tree` implemente: `insert(p,x)`, `find(x)` y `clear()`. Para la clase `iterator` implemente `operator!=`, `operator==`, `lchild()` y `right()` (u `operator++`). Declare además la clase `cell`. Observaciones:

- **Debe** declarar los miembros privados de las clases a declarar o implementar. Ayuda: use la figura 1.
- Si opta por la interfase *"estilo"* STL, implemente la forma **prefija** del operador `operator++` (`++p`).

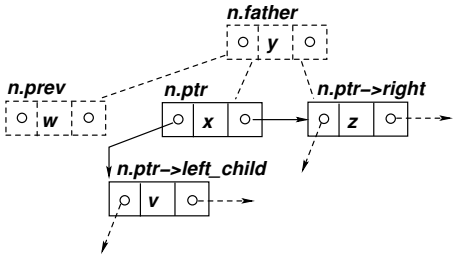

Figura 1: Entorno local de un iterator sobre árboles.

[Ej. 2] **[programación (total = 45 puntos)]**

a) **[longest-path (25 puntos)]**
Escribir una función `void longest_path(btree<int> &A,list<int> &L);` que, dado un árbol binario `A` retorna en `L` la lista de valores en el camino más largo en el árbol. Por ejemplo, si `A=(3 (4 2 .) (6 7 (8 9 5)))`, entonces `longest_path` debe retornar `L=(3 6 8 9)`. Si hay más de un camino con la longitud más larga, entonces puede retornar cualquiera de ellos. Por ejemplo, en el caso anterior puede retornar también `L=(3 6 8 5)`. *Notas:* se puede usar `list<int>::size()` y todas las funciones de lista en STL sin restricciones (asignación, constructor por copia...).

b) **[path-of-largest (20 puntos)]**
Escribir una función `void path_of_largest(tree<int> &A,list<int> &L);` que, dado un árbol ordenado orientado `A`, retorna en `L` el camino que se obtiene recorriendo el árbol hacia abajo, *siempre por el hijo más grande*. Por ejemplo, si `A=(3 (4 6 (7 8 9)) (5 2 4 6))` entonces `path_of_largest` debe retornar `L=(3 5 6)`. Si para un nodo el valor más grande de los hijos está repetido, entonces el camino puede seguir por cualquiera de ellos. Por ejemplo si `A=(3 (4 5 6) 4 2)` entonces `path_of_largest()` puede retornar `L=(3 4 6)` o `L=(3 4)`.

[Ej. 3] **[operativos (total = 25 puntos)]**

a) **[huffman (10 ptos)]** Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra `PAPELERAS`
$P(P) = 0.1, P(A) = 0.1, P(L) = 0.3, P(E) = 0.1, P(R) = 0.2, P(B) = 0.05, P(Q) = 0.05, P(S) = 0.1$
Calcular la longitud promedio del código obtenido. Justificar si cumple o no la condición de prefijos.

b) **[rec-arbol (5 ptos)]** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son

- `ORD_PRE` $=\{Z, K, Y, W, Q, M, N, T, U, V\}$;
- `ORD_POST` $=\{Y, W, K, M, V, U, T, N, Q, Z\}$.

c) [**construir-arbol (5 ptos)**] Escribir la secuencia de sentencias necesarias para construir el arbol binario `(3 (4 . 2) (1 . 7))`. (*Restricciones:* No usar `find()`).

d) [**modificar-arbol (5 ptos)**] Escribir la secuencia de secuencia necesarias para modificar el árbol ordenado orientado `A = (3 (4 5 6) (7 8 9))` de manera que se convierta en `A = (3 (4 5 6) (7 8 10 9))`. (Es decir, insertar el 10 entre el 8 y el 9). (*Restricciones:* No usar `find()`).

[Ej. 4] [Preguntas (total = 10 puntos, 2.5 puntos por pregunta)] Responder según el sistema *"multiple choice"*, es decir marcar con una cruz el casillero apropiado. **Atención:** Algunas respuestas son intencionalmente "descabelladas" y tienen puntajes **negativos**!!]

---

a) Dado el árbol `(a (b e f) (c (g h)) d))`, marcar cuáles de los siguientes son *caminos* válidos en el árbol.

- ☐ ... `(e b f)`
- ☐ ... `(f b a)`
- ☐ ... `(c g h)`
- ☐ ... `(a c g h)`
- ☐ ... `(d)`

---

b) ¿Cuál es el tiempo de ejecución para `splice(to,from)` en árboles binarios?

- ☐ ... $O(\log n)$
- ☐ ... $O(n)$
- ☐ ... $O(\sqrt{n})$
- ☐ ... $O(1)$

---

c) Dado el árbol `a=(1 3 (5 7 (9 11)))`, después de aplicar las siguientes sentencias:

```
n = a.find(5);
a.erase(n);
```

¿Como queda el árbol?

- ☐ ... `a=(1 3 (9 7 11))`
- ☐ ... `a=(1 3 (. 7 (9 11)))`
- ☐ ... `a=(1 3)`
- ☐ ... `a=(1 3 (7 (9 11)))`

---

d) Marcar cuáles de los siguientes son *árboles binarios completos* (ABC):

- ☐ ... `(1 2 (3 . 5))`
- ☐ ... `(1 2 3)`
- ☐ ... `(1 . (3 4 5))`
- ☐ ... `(1 2 (3 4 5))`
- ☐ ... `(1 (7 8 9) (3 4 5))`