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.
Examen Final. [10 de Febrero de 2005]

[Ej. 1] [**clases (20 puntos)**] Escribir el constructor por copia del TAD Arbol Ordenado Orientado (`class tree`) `tree<T>(const tree<T> &TT)`. Escribir las declaraciones de la clase y los componentes necesarios para implementar las funciones indicadas.

[Ej. 2] [**programacion (total = 80 puntos)**]

a) [**filter (total = 40 puntos)**] ... Escriba una función
`void filter(list<int> &L,bool (*pred)(int x));`
que elimina de la lista `L` aquellos elementos `x` que no satisfacen el predicado `pred(x)`. Por ejemplo, si `L=`$\{1,3,2,5,4,3,6,7\}$ y `bool impar(int x) { return x%2; }` , entonces `filter(L,impar)` debe dejar sólo los elementos impares, es decir `L=`$\{1,3,5,3,7\}$. *Restricciones:* Los elementos restantes deben quedar en el orden en el que estaban originalmente. El algoritmo debe ser $O(n)$, donde n es el número de elementos en la lista. El algoritmo debe ser *"in-place"*, es decir, no se debe usar ninguna estructura auxiliar.

b) [**cond-depth (total = 40 puntos)**] Implemente una función
`int cond_depth(tree<int> &T, ,bool (*pred)(int x));` que calcula la máxima profundidad de los elementos del árbol que satisfacen `pred()`. Por ejemplo, si `T=(1 (5 2 (9 6)) 3 (7 8))`, y `pred()` es `bool impar(int x) { return x%2; }` , entonces `cond_depth(T,impar)` debe retornar 2, ya que el elemento impar a mayor profundidad es el 9, que está a profundidad 2.

[Ej. 3] [**operativos (total=80pts) - LIBRES**]

a) [**ohash (20 ptos)**] Insertar los enteros (10,5,4,14,4,25,23,12,41,12,32,46) en una tabla de dispersión abierta con función de dispersión `h(x) = x % B` con `B`=10 cubetas. Mostrar como queda la tabla después de realizar las inserciones.

b) [**huffman (20 ptos)**] Dados los caracteres siguientes con sus correspondientes probabilidades, contruir el código binario y encodar la palabra `TSUNAMI`
$P(T) = 0.2, P(S) = 0.2, P(U) = 0.2, P(N) = 0.1, P(A) = 0.1, P(M) = 0.1, P(I) = 0.05, P(Z) = 0.05$
Calcular la longitud promedio del código obtenido.

c) [**heap (20 ptos)**] Dados los enteros {`7,1,6,2,5,3,4,10,12`} ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) antes y después de **cada** inserción/supresión.

d) [**reconstruir-arbol (20 puntos)**] Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son

- `ORD_PRE` $=\{A, Z, W, X, Y, T, R, Q, V, L\}$.
- `ORD_POST` $=\{Z, W, X, Y, R, Q, V, L, T, A\}$.

[Ej. 4] [**Preguntas (total = 20 puntos, 5puntos por pregunta) - LIBRES**] Algunas preguntas son según el sistema el sistema *"multiple choice"*, es decir marcar con una cruz el casillero apropiado. **Atención:** Algunas respuestas son intencionalmente "descabelladas" y tienen puntajes **negativos**!!]

---

a) Dadas las funciones

- $T_1(n) = 8n^3 + 2n + 3^{1/3}$, ‘
- $T_2(n) = \log 2 + 5n^4 + n!$,
- $T_3(n) = \sqrt{n} + n^{1/2} + n^{1/4} + \sqrt{5}n$ y
- $T_4(n) = (n+1)^2 + \log n$

ordenarlas de menor a mayor.

$$T_{\square} < T_{\square} < T_{\square} < T_{\square}$$

---

b) Sea el AOO `A=(5 7 (9 3 4) 6)`. Después de hacer las operaciones

```
p = A.find(9);
p = p.lchild(); p++;
p = A.insert(p,10);
```

¿ Cuál de las siguientes opciones es verdadera?

- ☐ Da un error al insertar.
- ☐ `A=(5 7 (9 3 4 10) 6)`, `*p=10`.
- ☐ `A=(5 7 (9 3  10 4) 6)`, `*p=10`.
- ☐ `A=(5 7 (9 3  10 4) 6)`, `*p=4`.

---

c) El tiempo de ejecución para el algoritmo de clasificación por montículos es

- $O(n \log n)$ en el mejor caso, $O(n^2)$ en promedio
- siempre $O(n \log n)$
- siempre $O(n^2)$
- a veces $O(n^2)$

---

d) Sea un tabla de dispersión abierta con B cubetas y n elementos. Asumiendo que la función de dispersión es lo suficientemente buena como para distribuir los elementos en forma uniforme entre las cubetas, el costo medio de inserción de un nuevo elemento es

- $O(n^2/B)$
- $O((n/B)^2)$
- $O(n + B)$
- $O(n/B)$