

Algoritmos y Estructuras de Datos. Recuperatorio Globalizador. [29 de Junio de 2006]

[Ej. 1] [Clases (20 pts)]

- **TAD's 1er Parcial:** Escribir la implementación en C++ del TAD LISTA (clase `list`) implementado por punteros ó cursores. Las funciones a implementar son `insert(p,x)`, `erase(p)`, `next()/iterator::operator++(int)`, `clear()`.
- **TAD's 2do Parcial:** Escribir la implementación en C++ del TAD Arbol Ordenado Orientado (clase `tree`) implementado con celdas enlazadas por punteros. Las funciones a implementar son `insert(p,x)`, `retrieve(p)`, `erase(p)`, `begin()`, `end()`, `clear()`. No es necesario implementar las clases `iterator` y `cell`.
- **TAD's 3er Parcial:** Escribir la implementación en C++ de las siguientes funciones del TAD conjunto por vectores de bits (clase `set`): `insert(x)`, `find(x)`, `erase(x)`, `set_difference(set &A, set &B, set &C)`

[Ej. 2] [Programación (total = 50 pts)]

- a) **[purga (10 pts)]**
 Escribir una función `void purga(list <T> & L)` que purga los elementos repetidos de una lista usando un conjunto como estructura auxiliar. Si un elemento está repetido varias veces, entonces en la lista purgada *debe quedar sólo la primera ocurrencia*. Por ejemplo, si $L = (1,5,7,3,2,4,3,1,7,1,3,9)$ entonces, después de hacer `purga(L)` debe quedar $L = (1,5,7,5,2,4,9)$. *Restricciones:* El algoritmo debe ser estable es decir que los elementos que quedan deben mantener la misma posición relativa. Además, el algoritmo debe ser $O(n)$.
- b) **[es-completo (20 pts)]**
 Escribir una función predicado `bool es_completo(btree<int> &)`, la cual retorna verdadero si el árbol binario es completo.
- c) **[grado (20 pts)]**
 Escribir una función `void degree(map<string, set<string> > &G, map<string, int> &dgr)`; que dado un grafo simétrico G cuyos vértices están identificados por strings retorna por `dgr` el grado de cada nodo (número de aristas conectadas que contienen al vértice).

[Ej. 3] [operativos (total = 20 pts)]

- a) **[lisp (5 pts)]:** Dibujar los árboles cuyas notaciones Lisp son:
 - Arbol ordenado orientado: `(a (b d (e f g h)) (c z))`
 - Arbol binario: `(z d (k . (j (l m n) .)))`
- b) **[arbol-expr (5 pts)]** Escribir el árbol que corresponde a la expresión $(a+b)/(x*y)+z-d/r$. Dar la notación prefija y postfija.
- c) **[ohash (5 pts)]** Insertar los enteros (15,10,9,19,9,29,28,17,46) en una tabla de dispersión abierta con función de dispersión $h(x) = x \% B$ con $B=10$ cubetas.
- d) **[rec-arbol (5 pts)]** Dibujar el árbol ordenado orientado cuyos nodos, listados en orden previo y posterior son
 - `ORD_PRE = {A, Z, X, Y, M, W, R, T, U, V}`,
 - `ORD_POST = {Y, M, X, Z, T, U, V, R, W, A}`.

- [Ej. 4] **[Preguntas (total = 10 pts, 2.5pts 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!!**

Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

- a) Dada la lista $L=(1,5,7,9,4)$. Cuál de las proposiciones es válida después de hacer las operaciones

```
q = L.begin();  
p=q++; p++; q++;  
p = L.erase(p);
```

- ☐ ... $*p=1$ y q es inválido.
☐ ... $*p=7$ y $*q=7$.
☐ ... $*p=7$ y q es inválido.
☐ ... $*p=9$ y $*q=5$.

- b) Dado el AOO $A=(1\ 5\ (3\ 4\ 2\ 7))$. Cuál de las proposiciones es válida después de hacer las operaciones

```
p = A.find(3);  
q = A.find(7);  
p = A.erase(p);  
p = A.insert(p,8);
```

- ☐ ... $A=(1\ 5\ (8\ 4\ 2\ 7))$ y q es inválido.
☐ ... $A=(1\ 5\ 8)$ y $*q=7$.
☐ ... $A=(1\ 5\ 8)$ y q es inválido.
☐ ... $A=(1\ 5\ (8\ 4\ 2\ 7))$ y $*q=7$.

- c) ¿Cuál listado aplicado a un árbol binario de búsqueda da los elementos ordenados?

- ☐ ... Orden previo.
☐ ... Orden simétrico.
☐ ... Orden posterior.
☐ ... Orden antisimétrico.

- d) ¿Cuál es el tiempo de ejecución de `splice(p,q)` para árboles?

- ☐ ... $O(1)$
☐ ... $O(n)$
☐ ... $O(1/n)$
☐ ... $O(\log_2 n)$